Journal of Artificial Intelligence Research 63 (2018) 51-86 Submitted 02/17; published 09/18

A Core Method for the Weak Completion Semantics with Skeptical Abduction

Emmanuelle-Anna Dietz Saldanha DIETZ@ICCL.TU-DRESDEN.DE
International Center for Computational Logic
TU Dresden 01062 Dresden, Germany

Steffen Hölldobler SH@ICCL.TU-DRESDEN.DE
International Center for Computational Logic
TU Dresden 01062 Dresden, Germany
and *North-Caucasus Federal University*
Stavropol, Russian Federation

Carroline Dewi Puspa Kencana Ramli CARROLINE.RAMLI@GMAIL.COM
International Center for Computational Logic
TU Dresden 01062 Dresden, Germany

Luis Palacios Medinacelli PALACIOS.MEDINACELLI@GMAIL.COM
LRASC, Thales Research & Technology, Palaiseau
and *LRI, CNRS, Université Paris-Saclay, France*

Abstract

The Weak Completion Semantics is a novel cognitive theory which has been successfully applied to the suppression task, the selection task, syllogistic reasoning, the belief bias effect, spatial reasoning as well as reasoning with conditionals. It is based on logic programming with skeptical abduction. Each program admits a least model under the three-valued Łukasiewicz logic, which can be computed as the least fixed point of an appropriate semantic operator. The semantic operator can be represented by a three-layer feed-forward network using the CORE method. Its least fixed point is the unique stable state of a recursive network which is obtained from the three-layer feed-forward core by mapping the activation of the output layer back to the input layer. The recursive network is embedded into a novel network to compute skeptical abduction. This paper presents a fully connectionist realization of the Weak Completion Semantics.

1. Introduction

In his seminal paper on the situation calculus, McCarthy (1963) formulated requirements for systems to reason about actions and causality. Besides being able to specify properties as formulas and to draw conclusions as logical consequences he suggested that *the formal descriptions of states should correspond as closely as possible to what people may reasonably be presumed to know about them when deciding what to do.* In order to meet this latter requirement we need to study humans and their behaviour, which is usually done within Cognitive Science.

In this paper, we are concerned with human reasoning tasks like, e.g., Byrne's (1989) suppression task, Wason's (1968) selection task, or syllogistic reasoning (Khemlani & Johnson-Laird, 2012). Firstly, we are interested in finding a declarative, computational logic approach adequately modeling these tasks. Secondly, we would like to embed the computational logic approach in a plausible connectionist or artificial neural network.

The Weak Completion Semantics (WCS) (Hölldobler, 2015) is a new cognitive theory which has been successfully applied to various human reasoning tasks WCS is rooted in the work by Stenning and van Lambalgen (2008) but corrects a technical bug by switching from three-valued Kripke-Kleene (Fitting, 1985) to three-valued Łukasiewicz (1920) logic (Hölldobler & Kencana Ramli, 2009a).

To illustrate WCS consider an example from the suppression task. Suppose we know that *if she has an essay to write then she will study late in the library.* Under WCS the given conditional is represented by the logic program

$$\mathcal{P}_1 = \{\ell \leftarrow e \wedge \neg ab_1,\ ab_1 \leftarrow \bot\},$$

where ℓ and e denote that *she will study late in the library* and *she has an essay to write*, respectively, and ab_1 is an abnormality predicate, which in this case is assumed to be false (considering the weak completion of the program). If we observe that *she will study late in the library*, then according to Byrne (1989), 71% of the participants concluded that *she has an essay to write.* This can be computed in WCS by abducing $\{e \leftarrow \top\}$ given $\mathcal{P}_1$.

Assume additionally, that *if she has a textbook to read then she will study late in the library.* Under WCS the two given conditionals are represented by the logic program

$$\mathcal{P}_2 = \mathcal{P}_1 \cup \{\ell \leftarrow t \wedge \neg ab_2,\ ab_2 \leftarrow \bot\},$$

where t denotes that *she has a textbook to read*, and ab_2 is another abnormality predicate, which is also assumed to be false. If we observe again that *she will study late in the library*, then the previously drawn conclusion is suppressed in that only 13% of the participants concluded that *she has an essay to write* (Byrne, 1989). If we apply abduction in WCS, then the observation can be explained by the two minimal explanations $\{e \leftarrow \top\}$ and $\{t \leftarrow \top\}$. Hence, reasoning credulously we would conclude that *she has an essay to write.* Apparently, humans don't do this; they appear to reason skeptically.

Programs like $\mathcal{P}_1$ and $\mathcal{P}_2$ as well as their weak completions admit a least model under WCS which can be computed as the least fixed point of an appropriate semantic operator (Hölldobler & Kencana Ramli, 2009a). Semantic operators for logic programs were first studied by Apt and van Emden (1982) in an attempt to capture the semantics of logic programs. In particular, they showed that in classical two-valued logic the least model of a definite logic program is identical to the least fixed point of a corresponding semantic operator. In the meantime, various semantic operators for more expressive logics have been studied.

In Cognitive Science semantic operators are interesting as they allow to construct a model for a given program. This construction may be compared to other ways of generating models like, for example, in the theory of mental models (Johnson-Laird & Byrne, 1991). Suppose we extend the program $\mathcal{P}_1$ with the fact $e \leftarrow \top$ representing the information that *she has an essay to write.* The extended program corresponds to the case of modus ponens

in the suppression task (Byrne, 1989). Starting with the empty interpretation, i.e., the interpretation where all relations are unknown, the semantic operator of WCS assigns *true* to e and *false* to ab_1 in its first application because of the fact $e \leftarrow \top$ and the negative assumption $ab_1 \leftarrow \bot$, respectively. In its second application, the operator additionally assigns *true* to ℓ because the condition of the rule

$$\ell \leftarrow e \wedge \neg ab_1$$

is *true* as soon as e is mapped to *true* and ab_1 is mapped to *false*. In other words, ℓ being *true* is an immediate consequence of the rule given that its condition is *true*. Further applications of the semantic operator do not alter the findings. A least fixed point has been reached. Reasoning with respect to this least fixed point allows to conclude ℓ, which is what 96% of the subjects in the suppression task do.

The semantic operator introduced by Apt and van Emden (1982) is continuous. Funahashi (1989) has shown that continuous mappings can be approximated arbitrary well by feed-forward networks. Combining both results, Hölldobler and Kalinke (1994) developed the idea to compute semantic operators for propositional logic programs by feed-forward networks. By connecting the output to the input layer, the feed-forward networks are turned into recurrent ones. These recurrent networks compute iterated applications of the semantic operators and, in particular, if they reach a stable state, then this state corresponds to the least fixed point of the semantic operator. In other words, the connectionist networks compute the least models of the given programs. The idea was later extended to first-order programs and called CORE method for connectionist model generation using recurrent networks with feed-forward core (Bader & Hölldobler, 2006).

The first question to be considered in this paper is: *How can the semantic operator associated with WCS be represented and computed within a fully connectionist setting?* This question will be answered in Section 3. But the solution is not just an extension of the CORE method to three-valued Łukasiewicz (1920) logic, but rather we extend the network to determine whether it has reached a stable state, to consider additional facts and assumptions in order to explain a given observation, to check whether a given set of integrity constraints is violated, and to eliminate stable states that may have arisen from previous reasoning episodes.

However, even with such a network we cannot solve all reasoning episodes of the suppression task. As discussed before, we have to add skeptical abduction. Unfortunately, all known connectionist solutions which we are aware of handle credulous abduction in classical two-valued logic (Ray & d'Avila Garcez, 2006; d'Avila Garcez, Gabbay, Ray, & Woods, 2007). Hence, the second question to be considered in this paper is: *How can skeptical abduction be integrated into the connectionist realization of the* CORE *method?* This question will be answered in Section 4. In particular, we will use McCulloch-Pitts (1943) networks to sequentially generate all possible explanations. Possible explanations are forwarded to the CORE network developed in Section 3. As soon as an explanation is detected, it will be stored. Once all possible explanations are tested, the skeptical conclusions will be computed.

In order to discuss the solutions to the two open research questions, WCS is presented in detail in the following Section 2. In particular, we will define logic programs, interpretations

and models, the weak completion semantics, its associated semantic operator, integrity constraints, and abductive frameworks.

Finally, in Section 5 we discuss our solutions and identify future research directions.

2. Weak Completion Semantics

In this section we describe the general notation that we use through the paper based on Lloyd's (1984) and Hölldobler's (2009) notation.

2.1 Logic Programs

An *atom* is an atomic proposition. A *literal* is either an atom A or its negation $\neg A$. A *(propositional logic) program* $\mathcal{P}$ is a finite set of clauses (or rules) of the form $A \leftarrow Body$, where the *head* A is an atom and $Body$ is either a non-empty conjunction of literals, $\top$ (denoting truth), or $\bot$ (denoting falsehood). Clauses of the form $A \leftarrow \top$ and $A \leftarrow \bot$ are called *(positive) facts* and *(negative) assumptions*, respectively. Let $atoms(\mathcal{P})$ denote the set of all atoms occurring in the program $\mathcal{P}$. An atom A is *defined* in $\mathcal{P}$ iff $\mathcal{P}$ contains a clause of the form $A \leftarrow Body$; otherwise A is said to be *undefined*. The set of all atoms that are defined in $\mathcal{P}$ is denoted by $def(\mathcal{P})$. The set of all atoms that are undefined in $\mathcal{P}$, i.e. $atoms(\mathcal{P}) \setminus def(\mathcal{P})$, is denoted by $undef(\mathcal{P})$.

As examples consider the programs

$$\mathcal{P}_3 = \{a \leftarrow b,\ a \leftarrow c,\ c \leftarrow \bot\}$$

and

$$\mathcal{P}_4 = \{c \leftarrow \top,\ c \leftarrow \bot\}.$$

We obtain

$$atoms(\mathcal{P}_3) = \{a, b, c\}, \quad def(\mathcal{P}_3) = \{a, c\}, \quad undef(\mathcal{P}_3) = \{b\}$$

and

$$atoms(\mathcal{P}_4) = def(\mathcal{P}_4) = \{c\}, \quad undef(\mathcal{P}_4) = \emptyset.$$

Consider the following transformation for a given program $\mathcal{P}$:

1. For all $A \in def(\mathcal{P})$, replace all clauses of the form $A \leftarrow Body_1,\ \ldots,\ A \leftarrow Body_n$ occurring in $\mathcal{P}$ by $A \leftarrow Body_1 \vee \ldots \vee Body_n$.
2. Replace all occurrences of $\leftarrow$ by $\leftrightarrow$.

The resulting set of equivalences is called the *weak completion* of $\mathcal{P}$ (Hölldobler & Kencana Ramli, 2009a), denoted by $wc\mathcal{P}$.

Returning to the examples $\mathcal{P}_3$ and $\mathcal{P}_4$ we obtain

$$wc\mathcal{P}_3 = \{a \leftrightarrow b \vee c,\ c \leftrightarrow \bot\}$$

and

$$wc\mathcal{P}_4 = \{c \leftrightarrow \top \vee \bot\}.$$

One should observe that the weak completion differs from the completion defined by Clark (1978) in that undefined atoms are not identified with falsehood. In particular, the completion of $\mathcal{P}_3$ is the set

$$wc\mathcal{P}_3 \cup \{b \leftrightarrow \bot\}.$$

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	$\top$

Table 1: Truth tables for the three-valued Łukasiewicz logic.

2.2 Interpretations and Models

A *(three-valued) interpretation* I is a mapping from the set of formulas (atoms, literals, clauses, programs, equivalences) into the set $\{\top, \bot, \mathsf{U}\}$ of truth values. It is represented by $\langle I^\top, I^\bot\rangle$ with the understanding that $I^\top$ and $I^\bot$ are the sets of all atoms mapped to $\top$ and $\bot$, respectively. Hence, $I^\top$ and $I^\bot$ are disjoint and $A \notin (I^\top \cup I^\bot)$ if and only if $I(A) = \mathsf{U}$. The meaning of the connectives $\neg$, $\wedge$, $\vee$, $\leftarrow$, and $\leftrightarrow$ is defined with respect to the Łukasiewicz (1920) (Ł-) logic and is given in Table 1.

One should observe that in contrast to two-valued logic, $a \leftarrow b$ is not semantically equivalent to $a \vee \neg b$ under Ł-logic: For interpretation I, where $I(a) = I(b) = \mathsf{U}$, we find $I(a \vee \neg b) = \mathsf{U}$ and $I(a \leftarrow b) = \top$. However, this is different under Kripke-Kleene logic (Kleene, 1952): For interpretation I, where $I(a) = I(b) = \mathsf{U}$, we find $I(a \vee \neg b) = \mathsf{U}$ and $I(a \leftarrow b) = \mathsf{U}$.

Returning to $\mathcal{P}_3$ again, we observe that the interpretation

$$I_1 = \langle\{b\}, \{a\}\rangle$$

maps $\mathcal{P}_3$ to $\bot$ because $I_1(b) = \top$, $I_1(a) = \bot$, and, consequently, $I_1(a \leftarrow b) = \bot$. On the other hand, the interpretation

$$I_2 = \langle\{a, b\}, \{c\}\rangle$$

maps $\mathcal{P}_3$ to $\top$ because $I_2(a \leftarrow b) = I_2(a \leftarrow c) = I_2(c \leftarrow \bot) = \top$.

An interpretation I for a formula F is said to be a *model* for F if and only if $I(F) = \top$. Hence, I_2 is a model for $\mathcal{P}_3$, and so are

$$I_3 = \langle\{a, b, c\}, \emptyset\rangle, \quad I_4 = \langle\emptyset, \{c\}\rangle, \quad I_5 = \langle\emptyset, \emptyset\rangle.$$

Hölldobler and Kencana Ramli (2009a) have shown that each logic program as well as its weak completion admits a least model with respect to the knowledge-ordering[1] under Ł-logic. It is defined as the intersection of all models for a given program, where

$$\langle I_1^\top, I_1^\top\rangle \cap \langle I_2^\top, I_2^\bot\rangle = \langle I_1^\top \cap I_2^\top, I_1^\bot \cap I_2^\bot\rangle.$$

Thus, I_5 is the least model for $\mathcal{P}_3$, whereas I_4 is the least model for $wc\mathcal{P}_3$ under Ł-logic (least Ł-model). Likewise, $\langle\{c\}, \emptyset\rangle$ is the least model for $wc\mathcal{P}_4$.

1. There are two common partial orderings on truth values, the *truth-* and the *knowledge-ordering*. In the former, the truth values are ordered such that $\bot \leq \mathsf{U} \leq \top$, whereas in the latter $\mathsf{U} \leq \top$ and $\mathsf{U} \leq \bot$ (e.g. Ruiz and Minker, 1995).

2.3 Weak Completion Semantics

In the sequel, let $\mathcal{M}_\mathcal{P}$ denote the least Ł-model of the program $\mathcal{P}$. We define logical consequence with respect to $\mathcal{M}_\mathcal{P}$, i.e.,

$$\mathcal{P} \models_{wcs} F \quad \text{if and only if} \quad \mathcal{M}_\mathcal{P}(F) = \top,$$

where $\mathcal{P}$ is a program and F a formula. This is called the *weak completion semantics* or WCS for short.

Returning to the examples $\mathcal{P}_3$ and $\mathcal{P}_4$ again, we find

$$\mathcal{P}_3 \models_{wcs} \neg c, \quad \mathcal{P}_3 \not\models_{wcs} a \vee \neg a, \quad \text{and} \quad \mathcal{P}_4 \models_{wcs} c.$$

The last example shows that under WCS positive facts are considered to be stronger than negative assumptions. This is also the reason why clauses of the forms $c \leftarrow \top$ and $c \leftarrow \bot$ are called facts and assumptions, respectively.

2.4 A Semantic Operator

The least Ł-model $\mathcal{M}_\mathcal{P}$ of a program $\mathcal{P}$ can be computed with the help of a semantic operator $\Phi_\mathcal{P}$ which was first introduced by Stenning and van Lambalgen (2008). Let I be an interpretation. The application of $\Phi_\mathcal{P}$ to I yields the interpretation $\langle J^\top, J^\bot \rangle$, where

$$\begin{array}{rcl} J^\top & = & \{A \mid \text{there exists a clause } A \leftarrow \mathit{Body} \in \mathcal{P} \text{ and } I(\mathit{Body}) = \top\}, \\ J^\bot & = & \{A \mid \text{there exists a clause } A \leftarrow \mathit{Body} \in \mathcal{P} \text{ and} \\ & & \quad \text{for all } A \leftarrow \mathit{Body} \in \mathcal{P} \text{ we find that } I(\mathit{Body}) = \bot\}. \end{array}$$

Kencana Ramli (2009) and Hölldobler (2009a) have shown that $\Phi_\mathcal{P}$ has a least fixed point which is identical to $\mathcal{M}_\mathcal{P}$ and can be computed by iterating $\Phi_\mathcal{P}$ starting with the empty interpretation.

Returning to the examples $\mathcal{P}_3$ and $\mathcal{P}_4$ we obtain

$$\Phi_{\mathcal{P}_3}(\langle \emptyset, \emptyset \rangle) = \langle \emptyset, \{c\} \rangle = \Phi_{\mathcal{P}_3}(\langle \emptyset, \{c\} \rangle) = \mathcal{M}_{\mathcal{P}_3}$$

and

$$\Phi_{\mathcal{P}_4}(\langle \emptyset, \emptyset \rangle) = \langle \{c\}, \emptyset \rangle = \Phi_{\mathcal{P}_4}(\langle \{c\}, \emptyset \rangle) = \mathcal{M}_{\mathcal{P}_4}.$$

Consider an extension of $\mathcal{P}_1$ where besides knowing that *if she has an essay to write then she will study late in the library* we also know that *she has an essay to write*:

$$\mathcal{P}_5 = \{\ell \leftarrow e \wedge \neg ab_1,\ ab_1 \leftarrow \bot,\ e \leftarrow \top\}.$$

We obtain

$$\begin{array}{lcl} \Phi_{\mathcal{P}_5}(\langle \emptyset, \emptyset \rangle) & = & \langle \{e\}, \{ab_1\} \rangle, \\ \Phi_{\mathcal{P}_5}(\langle \{e\}, \{ab_1\} \rangle & = & \langle \{e, \ell\}, \{ab_1\} \rangle \ = \ \Phi_{\mathcal{P}_5}(\langle \{e, \ell\}, \{ab_1\} \rangle \ = \ \mathcal{M}_{\mathcal{P}_5}, \end{array}$$

and conclude that *she will study late in the library*:

$$\mathcal{P}_5 \models_{wcs} \ell.$$

As another example consider

$$\mathcal{P}_6 = \{p \leftarrow q\}.$$

We obtain

$$\Phi_{\mathcal{P}_6}(\langle\emptyset,\emptyset\rangle) = \langle\emptyset,\emptyset\rangle = \mathcal{M}_{\mathcal{P}_6}.$$

The empty interpretation $\langle\emptyset,\emptyset\rangle$ is the least Ł-model of $\mathcal{P}_6$ computed by $\Phi_{\mathcal{P}_6}$ in accordance with the results presented by Hölldobler and Kencana Ramli (2009a). However, it is not a model under Kripke-Kleene logic (Kleene, 1952), where $\mathsf{U} \leftrightarrow \mathsf{U} = \mathsf{U}$. In fact, under Kripke-Kleene logic programs may not have least models with respect to the knowledge-ordering. In particular, $\mathcal{P}_6$ has two minimal models under Kripke-Kleene logic, viz. $\langle\{p,q\},\emptyset\rangle$ and $\langle\emptyset,\{p,q\}\rangle$, neither of which is computed by $\Phi_{\mathcal{P}_6}$. The corresponding results by Stenning and van Lambalgen (2008) are wrong.

2.5 Integrity Constraints

An *integrity constraint* is an expression of the form $\mathsf{U} \leftarrow \mathit{Body}$, where Body is a conjunction of literals. An interpretation I *violates* a finite set $\mathcal{IC}$ of integrity constraints if and only if $\mathcal{IC}$ contains an integrity constraint $\mathsf{U} \leftarrow \mathit{Body}$ with $I(\mathit{Body}) = \top$. Given an interpretation I and a set of integrity constraints $\mathcal{IC}$, I *satisfies* $\mathcal{IC}$ if and only if all clauses in $\mathcal{IC}$ are true under I.

2.6 Abductive Frameworks

A *(three-valued) abductive framework* $\langle\mathcal{P},\mathcal{A}_\mathcal{P},\mathcal{IC},\models_{wcs}\rangle$ consists of a program $\mathcal{P}$, a finite set

$$\mathcal{A}_\mathcal{P} = \{A \leftarrow \top \mid A \in \mathit{undef}(\mathcal{P})\} \cup \{A \leftarrow \bot \mid A \in \mathit{undef}(\mathcal{P})\}$$

of facts and assumptions called *abducibles*, a finite set $\mathcal{IC}$ of integrity constraints, and the logical consequence relation $\models_{wcs}$.

An *observation* $\mathcal{O}$ is a non-empty set of literals. $\mathcal{E} \subseteq \mathcal{A}_\mathcal{P}$ is an *explanation* for $\mathcal{O}$ given $\mathcal{P}$ and $\mathcal{IC}$ if and only if $\mathcal{P} \cup \mathcal{E} \models_{wcs} \mathcal{O} \cup \mathcal{IC}$. Sometimes, only minimal explanations are considered. A formula F *follows skeptically from* $\mathcal{P}$, $\mathcal{IC}$, *and* $\mathcal{O}$ if and only if $\mathcal{O}$ can be explained, and for all explanations $\mathcal{E}$ for $\mathcal{O}$ it holds that $\mathcal{P}\cup\mathcal{E} \models_{wcs} F$. F *follows credulously from* $\mathcal{P}$, $\mathcal{IC}$, *and* $\mathcal{O}$ if and only if there exists an explanation $\mathcal{E}$ for $\mathcal{O}$ such that $\mathcal{P}\cup\mathcal{E} \models_{wcs} F$.

Returning to the example $\mathcal{P}_1 = \{\ell \leftarrow e \wedge \neg ab_1,\ ab_1 \leftarrow \bot\}$ of the introductory Section 1 we learn that

$$\mathit{def}(\mathcal{P}_1) = \{\ell, ab_1\}, \quad \mathit{undef}(\mathcal{P}_1) = \{e\},$$

and, hence,

$$\mathcal{A}_{\mathcal{P}_1} = \{e \leftarrow \top,\ e \leftarrow \bot\}.$$

Let $\mathcal{IC} = \emptyset$ and $\mathcal{O} = \{\ell\}$. There are four possible subsets of $\mathcal{A}_{\mathcal{P}_1}$, viz.

$$\mathcal{E}_0 = \emptyset, \quad \mathcal{E}_1 = \{e \leftarrow \top\}, \quad \mathcal{E}_2 = \{e \leftarrow \bot\}, \quad \mathcal{E}_3 = \mathcal{A}_{\mathcal{P}_1}.$$

Combining these subsets with $\mathcal{P}_1$ and computing the corresponding least Ł-models we obtain

$$\begin{array}{lcll} \mathcal{M}_{\mathcal{P}_1\cup\mathcal{E}_0} & = & \langle\emptyset,\{ab_1\}\rangle, & \\ \mathcal{M}_{\mathcal{P}_1\cup\mathcal{E}_1} & = & \langle\{e,\ell\},\{ab_1\}\rangle & = \quad \mathcal{M}_{\mathcal{P}_5}, \\ \mathcal{M}_{\mathcal{P}_1\cup\mathcal{E}_2} & = & \langle\emptyset,\{ab_1,e,\ell\}\rangle, & \\ \mathcal{M}_{\mathcal{P}_1\cup\mathcal{E}_3} & = & \langle\{e,\ell\},\{ab_1\}\rangle & = \quad \mathcal{M}_{\mathcal{P}_1\cup\mathcal{E}_1}. \end{array}$$

All models satisfy $\mathcal{IC}$, whereas only the models $\mathcal{M}_{\mathcal{P}_1 \cup \mathcal{E}_1}$ and $\mathcal{M}_{\mathcal{P}_1 \cup \mathcal{E}_3}$ assign $\top$ to ℓ. Thus, $\mathcal{E}_1$ and $\mathcal{E}_3$ are explanations for $\mathcal{O}$. Because $\mathcal{E}_1 \subset \mathcal{E}_3$, the explanation $\mathcal{E}_1$ is minimal. We conclude (skeptically as well as credulously) that *she has an essay to write*.

It is no coincidence that $\mathcal{M}_{\mathcal{P}_1 \cup \mathcal{E}_1} = \mathcal{M}_{\mathcal{P}_1 \cup \mathcal{E}_3}$. A pair of clauses of the form $c \leftarrow \top$ and $c \leftarrow \bot$ is said to be *complementary*. A set of clauses is said to be *complementary* if it contains a complementary pair. In the example discussed in the previous paragraph, $\mathcal{E}_3$ is complementary with complementary pair $e \leftarrow \top$ and $e \leftarrow \bot$.

Proposition 1. *Let $\langle \mathcal{P}, \mathcal{A}_\mathcal{P}, \mathcal{IC}, \models_{wcs} \rangle$ be an abductive framework, $\mathcal{O}$ an observation, and $\mathcal{E} \subseteq \mathcal{A}_\mathcal{P}$ an explanation for $\mathcal{O}$ which contains a complementary pair $c \leftarrow \top$ and $c \leftarrow \bot$. Then, $\mathcal{E}' = \mathcal{E} \setminus \{c \leftarrow \bot\}$ is also an explanation for $\mathcal{O}$ and $\mathcal{M}_{\mathcal{P} \cup \mathcal{E}} = \mathcal{M}_{\mathcal{P} \cup \mathcal{E}'}$.*

Proof If $\mathcal{E}$ contains $c \leftarrow \top$ and $c \leftarrow \bot$, then $c \in \mathit{undef}(\mathcal{P})$. Hence, $wc(\mathcal{P} \cup \mathcal{E})$ contains the equivalence $c \leftrightarrow \top \vee \bot$. This is semantically equivalent to $c \leftrightarrow \top$, which is the corresponding equivalence contained in $wc(\mathcal{P} \cup \mathcal{E}')$ where $\mathcal{E}' = \mathcal{E} \setminus \{c \leftarrow \bot\}$. The result follows from the observation that all other equivalences occurring in $wc(\mathcal{P} \cup \mathcal{E})$ and $wc(\mathcal{P} \cup \mathcal{E}')$ are identical. ■

As an immediate consequence of Proposition 1, we learn that explanations can be restricted to non-complementary ones.

Proposition 2. *Given n undefined atoms in $\mathcal{P}$, there are 2^{2^n} subsets of $\mathcal{A}_\mathcal{P}$ and 3^n non-complementary subsets of $\mathcal{A}_\mathcal{P}$.*

Proof As $|\mathcal{A}_\mathcal{P}| = 2 \times n$, it follows straightforwardly that there are 2^{2^n} subsets of $\mathcal{A}_\mathcal{P}$. It can be easily proven by induction on n that there are 3^n non-complementary subsets of $\mathcal{A}_\mathcal{P}$. The case $n = 1$ has been discussed in the example above. Suppose that the result holds for n, the program $\mathcal{P}$ has $n+1$ undefined atoms, and c is one of the undefined atoms occurring in $\mathcal{P}$. Let $\mathcal{A}'_\mathcal{P} = \mathcal{A}_\mathcal{P} \setminus \{c \leftarrow \top, c \leftarrow \bot\}$. By the induction hypothesis $\mathcal{A}'_\mathcal{P}$ has 3^n non-complementary subsets. To each of the subsets we can add either $\emptyset$, $\{c \leftarrow \top\}$, or $\{c \leftarrow \bot\}$ to obtain the non-complementary subsets of $\mathcal{A}_\mathcal{P}$. Altogether, these are $3 \times 3^n = 3^{n+1}$ subsets. An application of the induction principle completes the proof. ■

Considering the example $\mathcal{P}_2 = \mathcal{P}_1 \cup \{\ell \leftarrow t \wedge \neg ab_2,\ ab_2 \leftarrow \bot\}$ of the Section 1 we learn that

$$\mathit{def}(\mathcal{P}_2) = \{\ell, ab_1, ab_2\}, \quad \mathit{undef}(\mathcal{P}_2) = \{e, t\},$$

$n = 2$, and,

$$\mathcal{A}_{\mathcal{P}_2} = \{e \leftarrow \top,\ t \leftarrow \top,\ e \leftarrow \bot,\ t \leftarrow \bot\}.$$

Let $\mathcal{IC} = \emptyset$ and $\mathcal{O} = \{\ell\}$. There are nine non-complementary subsets of $\mathcal{A}_{\mathcal{P}_2}$, viz.

$$\begin{array}{ll}
\mathcal{E}_0 = \emptyset, & \mathcal{E}_1 = \{e \leftarrow \top\}, \\
\mathcal{E}_2 = \{e \leftarrow \bot\}, & \mathcal{E}_3 = \{t \leftarrow \top\}, \\
\mathcal{E}_4 = \{t \leftarrow \bot\}, & \mathcal{E}_5 = \{e \leftarrow \top, t \leftarrow \top\}, \\
\mathcal{E}_6 = \{e \leftarrow \top, t \leftarrow \bot\}, & \mathcal{E}_7 = \{e \leftarrow \bot, t \leftarrow \top\}, \\
\mathcal{E}_8 = \{e \leftarrow \bot, t \leftarrow \bot\}. &
\end{array}$$

Combining these subsets with $\mathcal{P}_2$ and computing the corresponding least Ł-models we obtain

$$\begin{aligned}
\mathcal{M}_{\mathcal{P}_2\cup\mathcal{E}_0} &= \langle \emptyset, \{ab_1, ab_2\}\rangle, && (1)\\
\mathcal{M}_{\mathcal{P}_2\cup\mathcal{E}_1} &= \langle \{e, \ell\}, \{ab_1, ab_2\}\rangle, && (2)\\
\mathcal{M}_{\mathcal{P}_2\cup\mathcal{E}_2} &= \langle \emptyset, \{e, ab_1, ab_2\}\rangle, &&\\
\mathcal{M}_{\mathcal{P}_2\cup\mathcal{E}_3} &= \langle \{t, \ell\}, \{ab_1, ab_2\}\rangle, && (3)\\
\mathcal{M}_{\mathcal{P}_2\cup\mathcal{E}_4} &= \langle \emptyset, \{t, ab_1, ab_2\}\rangle, &&\\
\mathcal{M}_{\mathcal{P}_2\cup\mathcal{E}_5} &= \langle \{e, t, \ell\}, \{ab_1, ab_2\}\rangle, && (4)\\
\mathcal{M}_{\mathcal{P}_2\cup\mathcal{E}_6} &= \langle \{e, \ell\}, \{t, ab_1, ab_2\}\rangle, && (5)\\
\mathcal{M}_{\mathcal{P}_2\cup\mathcal{E}_7} &= \langle \{t, \ell\}, \{e, ab_1, ab_2\}\rangle, && (6)\\
\mathcal{M}_{\mathcal{P}_2\cup\mathcal{E}_8} &= \langle \emptyset, \{e, t, ab_1, ab_2\}\rangle. && (7)
\end{aligned}$$

There are five explanations, viz. $\mathcal{E}_1$, $\mathcal{E}_3$, $\mathcal{E}_5$, $\mathcal{E}_6$, $\mathcal{E}_7$, with $\mathcal{E}_1$ and $\mathcal{E}_3$ being the minimal ones. We cannot conclude skeptically that *she has an essay to write*, because only $\mathcal{M}_{\mathcal{P}_2\cup\mathcal{E}_1}$ maps e to $\top$, whereas $\mathcal{M}_{\mathcal{P}_2\cup\mathcal{E}_3}$ maps e to U. This is in line with the findings presented by Byrne (1989) and shows that skeptical abduction is needed to adequately model the suppression task. We will come back to this example in Subsection 4.2.

3. A Core Method for the Computation of Least Ł-Models

Hölldobler, Kalinke, Hizler and Seda presented a connectionist model generator for propositional logic programs using recurrent networks with a feed-forward core, i.e, a feed-forward network consisting of input, hidden and output layer, of logical threshold units (Hölldobler & Kalinke, 1994; Hitzler, Hölldobler, & Seda, 2004). The input and output layers of the feed-forward core were used to represent interpretations. The feed-forward network itself was used to compute the application of a semantic operator associated with a given logic program. This operator is applied to the interpretation given by the activation pattern of the input layer at time point t and yields the interpretation given by the activation pattern of the output layer at time point $t + 2$, i.e., the feed-forward network needs two time steps to propagate the activation pattern of the input layer to the output layer. The recurrent connections between the output and the input layer of the feed-forward core were used to iterate the application of the semantic operator. If a stable state was reached then the activation pattern of the input (or output) layer represent the least fixed point of the semantic operator. The connectionist model generator was later called the CORE method (Bader & Hölldobler, 2006).

The CORE method has been extended and applied to a variety of programs and logics. It was applied to reflexive reasoning (Hölldobler, Kalinke, & Wunderlich, 2000). The logical threshold units can be replaced by squashing units (d'Avila Garcez, Zaverucha, & de Carvalho, 1997; Bader, 2009). Consequently, the semantic operator computed by the feed-forward core can be learned by back-propagation. Modal, intuitionistic, and temporal logic programs were considered (d'Avila Garcez, Lamb, & Gabbay, 2002, 2003, 2009). The CORE method was extended to first-order logic programs (Hölldobler, Kalinke, & Störr, 1999; Bader, Hitzler, & Hölldobler, 2008) based on the idea that feed-forward connectionist networks can approximate almost all functions arbitrarily well (Hornik, Stinchcombe, &

Figure 1: The output v of the unit u_1 is multiplied with the weight ω and received as the input i of the unit u_2 if the modifier (or output) m of unit u_3 is 1 (left; positively modified connection) or 0 (right; negatively modified connection), respectively.

White, 1989; Funahashi, 1989) and, hence, they can also approximate the semantic operators associated with first-order logic programs.

Kalinke (1994) has applied the CORE method to propositional logic programs under the three-valued Kripke-Kleene logic used by Fitting (1985). In particular, her feed-forward cores compute a semantic operator introduced by Fitting. This operator differs from the $\Phi_{\mathcal{P}}$ operator presented in Subsection 2.4 in that in the definition of $J^{\perp}$ the condition '*there exists a clause* $A \leftarrow$ *Body*' is omitted; consequently, Fitting's operator computes a minimal model for the completion of a given program. Seda and Lane (2004) showed that the CORE method can be extended to many-valued logic programs using again the three-valued Kripke-Kleene logic and Fitting's operator. In the sequel, the approaches by Kalinke as well as by Seda and Lane are modified in order to compute $\Phi_{\mathcal{P}}$.

3.1 Some Preliminaries about Connectionist Systems

We assume the reader to be familiar with connectionist networks as, for example, defined by Feldman and Ballard (1982). In particular, we will use McCulloch-Pitts (1943) networks of threshold units and multi-layer feed-forward networks (Rumelhart, Hinton, & McClelland, 1986). We sometimes use modified connections of the following form: In case of a *positively modified connection*, the input $i = wv$ of a unit is only received if the modifier m is 1, where v is the output of the sending unit and w is the weight of the connection:

$$i = \begin{cases} wv & \text{if } m = 1, \\ 0 & \text{if } m = 0. \end{cases}$$

In case of a *negatively modified connection*, the input $i = wv$ is only received if the modifier m is 0:

$$i = \begin{cases} 0 & \text{if } m = 1, \\ wv & \text{if } m = 0. \end{cases}$$

The modifier m is the output of another unit. Figure 1 shows the graphical representation of a positively and a negatively modified connection.

3.2 A Translation Algorithm

Given a program $\mathcal{P}$, the following algorithm translates $\mathcal{P}$ into a feed-forward core $\mathcal{N}_{\mathcal{P}}$. Let $m = |atoms(\mathcal{P})|$ 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. Let $i \in [1, m]$. The $2i-1$-st unit in the layers, denoted by $i^\top$, is active if and only if the i-th variable is mapped to $\top$. The $2i$-th unit in the layers, denoted by $i^\bot$, is active if and only if the i-th variable is mapped to $\bot$. Both, the $2i-1$-st and the $2i$-th unit, are passive if and only if the i-th variable is mapped to U. The case where the $2i-1$-st and the $2i$-th unit are active is not allowed.

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

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

2. For each fact $A \leftarrow \top$ occurring in $\mathcal{P}$, do the following:

   (a) Add two binary threshold units $h^\top$ and $h^\bot$ to the hidden layer.
   Set the threshold of both units to $\frac{\omega}{2}$.

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

   (c) Connect the unit $\top$ in the input layer to $h^\top$.

3. For each assumption $A \leftarrow \bot$ occurring in $\mathcal{P}$, do the following:

   (a) Add two binary threshold units $h^\top$ and $h^\bot$ to the hidden layer.
   Set the threshold of both units to $\frac{\omega}{2}$.

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

   (c) Connect the unit $\bot$ in the input layer to $h^\bot$.

4. 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 $h^\bot$ to the hidden layer.

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

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

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

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

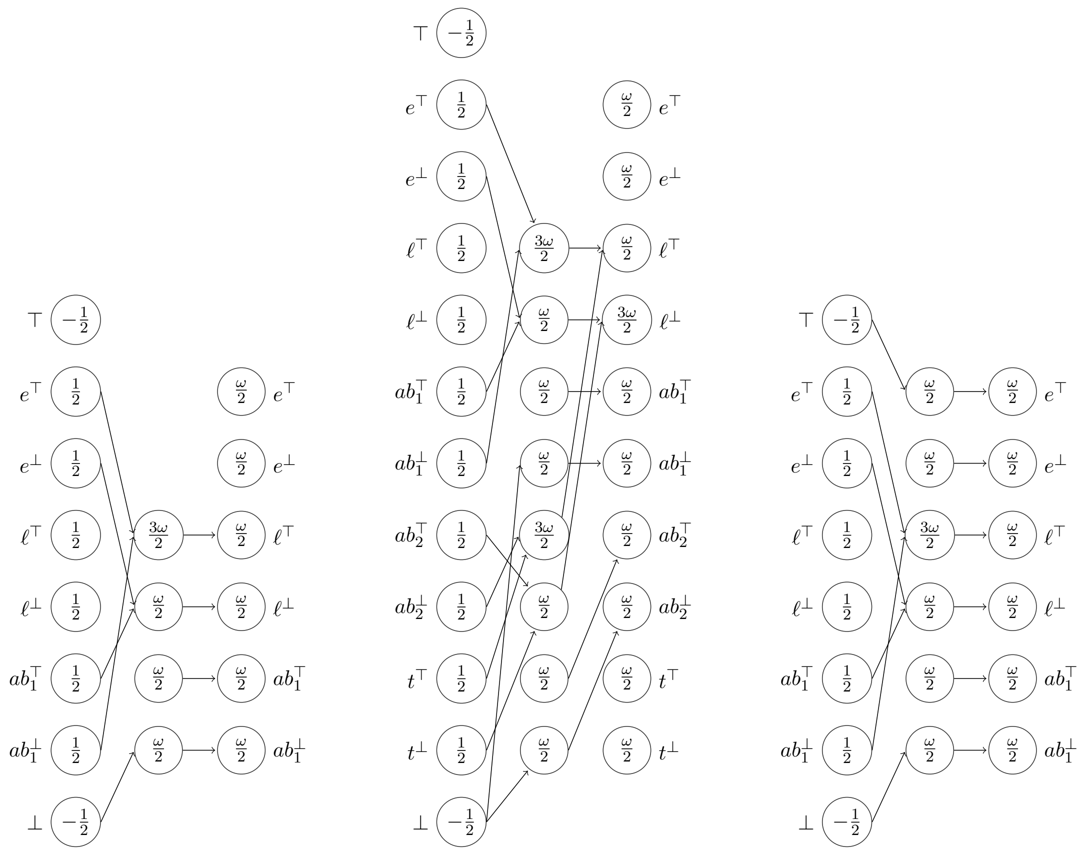

Figure 2: The cores $\mathcal{N}_{\mathcal{P}_1}$ (left), $\mathcal{N}_{\mathcal{P}_2}$ (middle), and $\mathcal{N}_{\mathcal{P}_5}$ (right). All connections have the weight ω. The units representing $\top$ and $\bot$ have been added at the top and the bottom of the input layer.

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

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

As examples reconsider $\mathcal{P}_1$, $\mathcal{P}_2$, and $\mathcal{P}_5$ discussed in the introduction as well as in Subsection 2.4. The corresponding networks are shown in Figure 2.

3.3 Properties

Proposition 3. *For each program $\mathcal{P}$, there exists a core $\mathcal{N}_\mathcal{P}$ of binary threshold units computing $\Phi_\mathcal{P}$.*

Proof Let $\mathcal{P}$ be a program, $\mathcal{N}_\mathcal{P}$ the core obtained by the translation algorithm, and assume that the input layer is activated at time t such that it represents an interpretation I. At time $t+1$ the following units of the hidden layer become active:

- An $h^\top$-unit representing $A \leftarrow \top$ becomes active, because the unit representing $\top$ in the input layer has negative threshold, receives no input and, thus, is active, the sum of the weighted input of the unit $h^\top$ is ω and is larger than its threshold of $\frac{\omega}{2}$.
- An $h^\top$- unit representing $A \leftarrow B_1 \wedge \ldots \wedge B_k$ becomes active if and only if all units representing $B_1, \ldots, B_k$ in the input layer are active, i.e., if $I(B_1) = \ldots = I(B_k) = \top$, and, thus, the sum of the weighted input of $h^\top$ is $k\omega$ and is greater than its threshold of $k\omega - \frac{\omega}{2}$.
- An $h^\bot$-unit representing $A \leftarrow \bot$ becomes active, because the unit representing $\bot$ in the input layer has negative threshold, receives no input and, thus, is active, the sum of the weighted input of the unit $h^\bot$ is ω and is larger than its threshold of $\frac{\omega}{2}$.
- An $h^\bot$-unit representing $A \leftarrow B_1 \wedge \ldots \wedge B_k$ in the hidden layer becomes active if and only if at least one unit representing the negation of $B_1, \ldots, B_k$ in the input layer is active, i.e., if $I(\neg B_1) \vee \ldots \vee I(\neg B_k) = \top$, and, thus, the sum of the weighted input of $h^\bot$ is greater or equal than ω, which is greater than its threshold of $\frac{\omega}{2}$.

At time $t+2$ the following units in the output layer become active:

- A unit representing A becomes active if and only if there is an active $h^\top$-unit representing $A \leftarrow \mathit{Body}$ in the hidden layer at time $t+1$ and, thus, the sum of the weighted input of the unit representing A is larger or equal than ω, which is larger than its threshold of $\frac{\omega}{2}$.
- A unit representing $\neg A$ becomes active if and only if all $h^\bot$-units representing a clause with head A in the hidden layer are active at time $t+1$ and there is at least one such $h^\bot$ unit. Suppose $l = |\mathit{def}(A, \mathcal{P})|$, then $l \geq 1$ and each of the l $h^\bot$-units must be active at time $t+1$. In this case, the sum of the weighted input of the unit representing $\neg A$ is $l\omega$ which is larger than its threshold of $max\{\frac{\omega}{2}, l\omega - \frac{\omega}{2}\}$.

Hence, at time $t+2$ the activation pattern of the output layer represents $\Phi_\mathcal{P}(I)$. ■

Using the techniques presented by d'Avila Garcez et al. (1997) and Bader (2009) the logical threshold units can be replaced by sigmoidal ones such that the core can be trained by back-propagation.

Given a program $\mathcal{P}$ and its core $\mathcal{N}_\mathcal{P}$, a recurrent network $\mathcal{N}_\mathcal{P}^\circlearrowleft$ can be constructed by connecting each unit in the output layer to its corresponding unit in the input layer with weight 1. In Figure 3 the construction is illustrated for the programs $\mathcal{P}_1$ and $\mathcal{P}_5$.

Proposition 4. *For each program $\mathcal{P}$, the corresponding recurrent network $\mathcal{N}_\mathcal{P}^\circlearrowleft$ initialized by the empty interpretation will converge to a stable state which corresponds to the least fixed point of $\Phi_\mathcal{P}$.*

Proof The result follows immediately from the construction of the recurrent network using Proposition 3 and that $\Phi_\mathcal{P}$ has a least fixed point which can be computed by iterating $\Phi_\mathcal{P}$ starting with the empty interpretation, as has been shown by Hölldobler and Kencana Ramli (2009a). ■

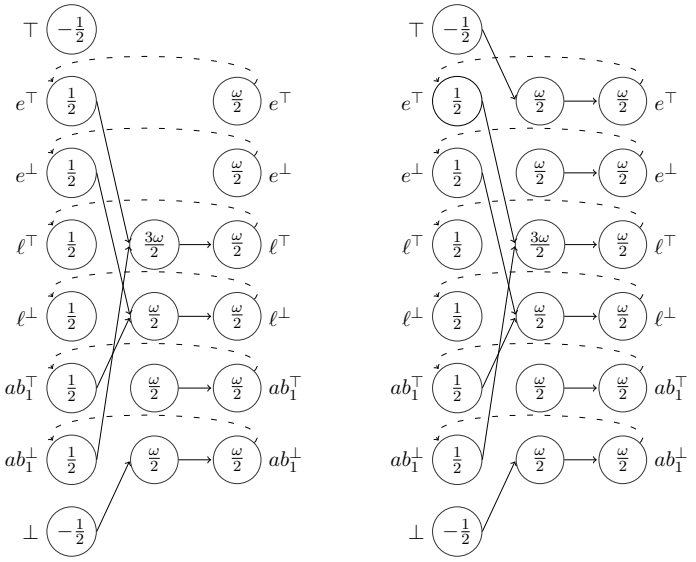

Figure 3: The recurrent networks $\mathcal{N}^{\circlearrowleft}_{\mathcal{P}_1}$ (left) and $\mathcal{N}^{\circlearrowleft}_{\mathcal{P}_5}$ (right). All dashed connections have the weight 1; all other connections have the weight ω.

In Figure 4 the computation is exemplified using the program $\mathcal{P}_5$. Starting at $t = 0$ with the empty network as shown in Figure 3, at $t = 1$ the units $\top$ and $\bot$ in the input layer become active. At $t = 2$, the units in the hidden layer which are corresponding to the fact $e \leftarrow \top$ and the assumption $ab_1 \leftarrow \bot$ become active. At $t = 3$ the units $e^\top$ and $ab_1^\bot$ in the output layer become active. This activation is propagated to the input layer at $t = 4$. At $t = 5$ the $h^\top$-unit of the hidden layer and corresponding to the clause $\ell \leftarrow e \wedge \neg ab_1$ becomes active, which leads to the activation of the unit $\ell^\top$ in the output layer at $t = 6$. Finally at $t = 7$ this is propagated to the input layer leading to a stable state which represents the least Ł-model $\langle\{e, \ell\}, \{ab_1\}\rangle$ of $\mathcal{P}_5$ as the activation pattern of the input or output layer.

Figure 5 shows the computation of the least Ł-model of the program $\mathcal{P}_1$. Starting at $t = 0$ with the empty network as shown in Figure 2, at $t = 1$ the units $\top$ and $\bot$ in the input layer become active. At time $t = 2$, the $h^\bot$-unit representing $ab_1 \leftarrow \bot$ in the hidden layer becomes active. At $t = 3$ the unit $ab_1^\bot$ in the output layer becomes active. At $t = 4$ this is propagated to the input layer leading to the unique stable state representing the least Ł-model $\langle\emptyset, \{ab_1\}\rangle$ of $\mathcal{P}_1$ as the activation pattern of the input or output layer.

3.4 Clamping Units of the Output Layer

Reconsider the program $\mathcal{P}_1$. Suppose we are making the observation $\mathcal{O} = \{\ell\}$. This observation cannot be explained by the least Ł-model $\langle\emptyset, \{ab_1\}\rangle$ of $\mathcal{P}_1$ or, equivalently, by the stable state of the corresponding recurrent network $\mathcal{N}^{\circlearrowleft}_{\mathcal{P}_1}$ shown in Figure 5. As discussed in Subsection 2.6, the two minimal and non-complementary candidates for explanations for $\mathcal{O}$ with respect to the abductive framework $\langle\mathcal{P}_1, \mathcal{A}_{\mathcal{P}_1}, \emptyset, \models_{wcs}\rangle$ are $\{e \leftarrow \top\}$ and $\{e \leftarrow \bot\}$. In order to test whether these candidate sets explain $\mathcal{O}$ we could construct recurrent networks

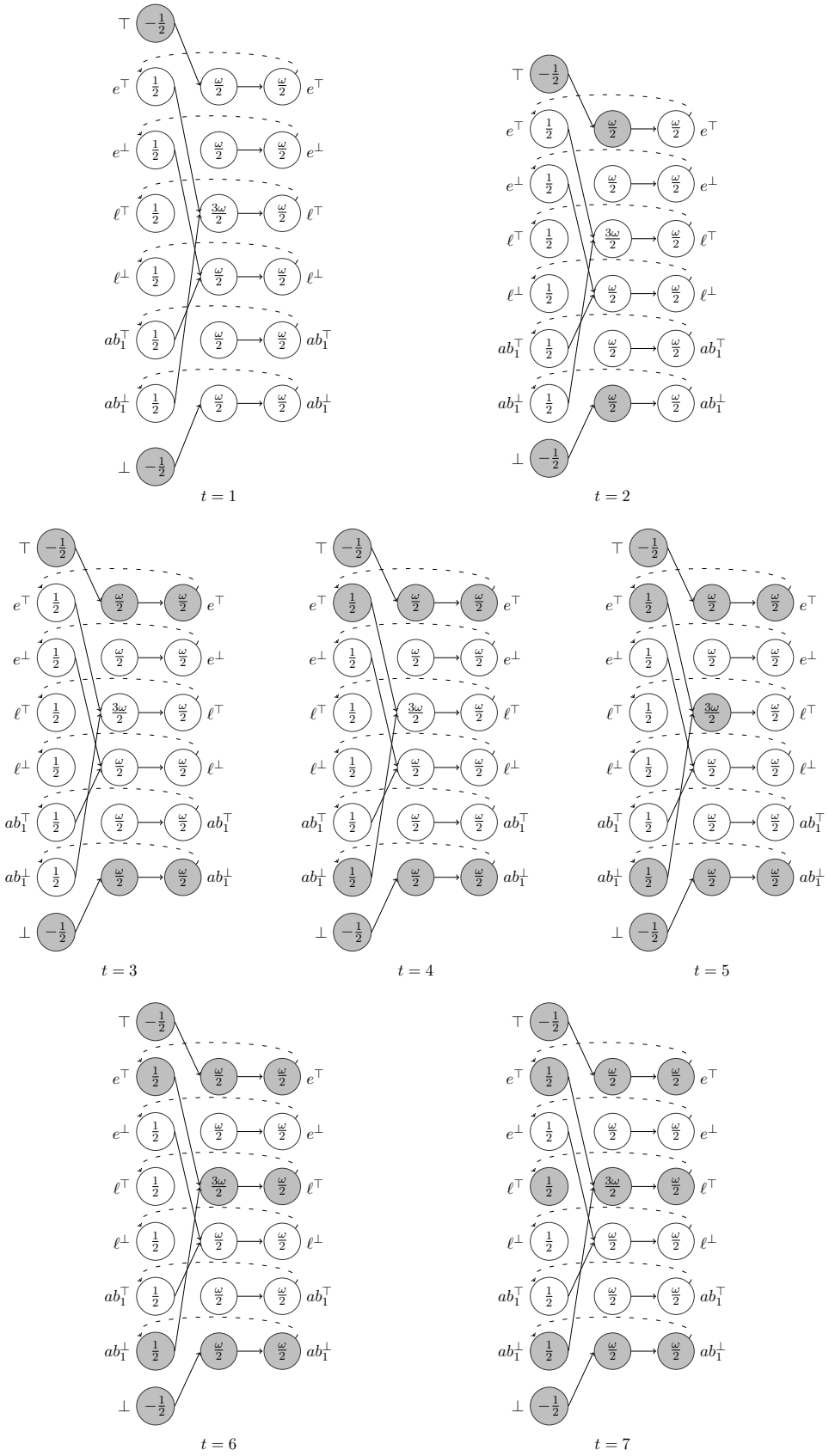

Figure 4: The computation of the unique stable state for the recurrent network $\mathcal{N}^{\circlearrowleft}_{\mathcal{P}_5}$.

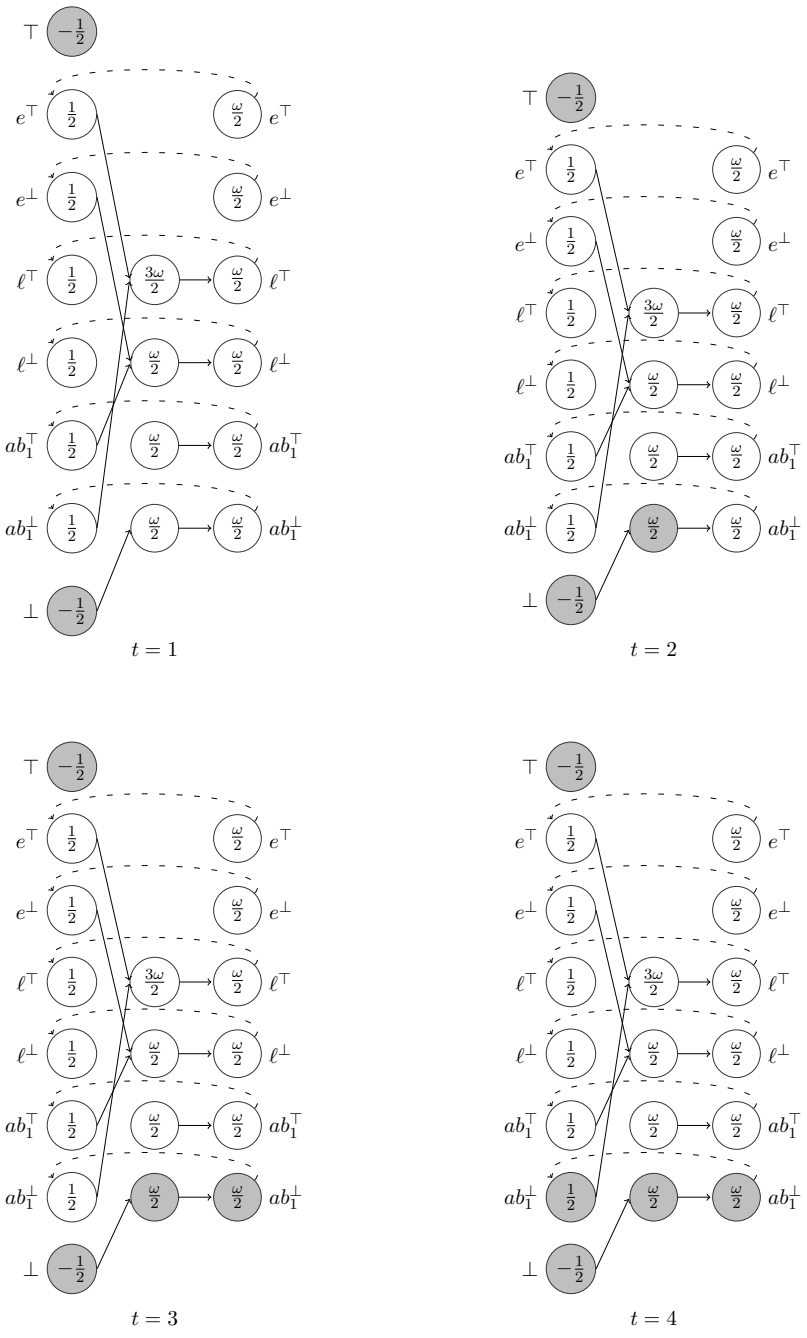

Figure 5: The computation of the unique stable state for the recurrent network $\mathcal{N}^{\heartsuit}_{\mathcal{P}_1}$, where active units are depicted in grey.

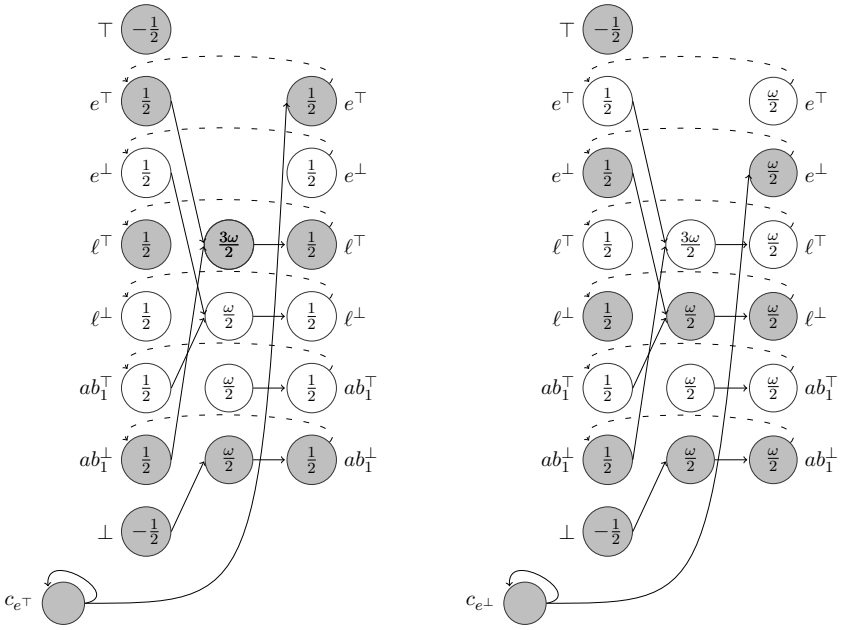

Figure 6: The stable states of the recurrent network for $\mathcal{P}_1$ with externally clamping the unit $e^\top$ in the output layer (left) and the unit $e^\bot$ in the output layer (right).

for $\mathcal{P}_5 = \mathcal{P}_1 \cup \{e \leftarrow \top\}$ and $\mathcal{P}_1 \cup \{e \leftarrow \bot\}$ and check whether their stable states contain an active unit representing ℓ in their input and output layers.

But, instead of adding hidden layer units and their connections for these candidate sets of facts to the network representing $\mathcal{P}_1$ we could simply clamp the units $e^\top$ and $e^\bot$ in the output layer. Figure 6 shows the computations of the networks, where the units $e^\top$ and $e^\bot$ in the output layer are externally clamped via the units $c_{e^\top}$ and $c_{e^\bot}$, respectively. Now, the stable states encoding the least Ł-models $\langle\{e, \ell\}, \{ab_1\}\rangle$ of $(\mathcal{P}_1 \cup \{e \leftarrow \top\})$ and $\langle\emptyset, \{ab_1, e, \ell\}\rangle$ of $\mathcal{P}_1 \cup \{e \leftarrow \bot\})$ are computed. One should observe that the units $e^\top$ and $e^\bot$ are not simultaneously clamped. Due to Proposition 1 possible explanations are non-complementary and, thus, cannot contain a pair $e \leftarrow \top, e \leftarrow \bot$. This is also in line with the representation $\langle I^\top, I^\bot\rangle$ of interpretations, where it is required that $I^\top \cap I^\bot = \emptyset$.

The units $c_{e^\top}$ and $c_{e^\bot}$ are self-excitatory such that they remain active once they have been externally activated. This external activation will be provided by the network $\mathcal{N}_\mathcal{A}$ discussed in Section 4.1. As discussed in Section 3.5 the self-excitatory connections are in fact modified such that the external activation can be withdrawn once the stable state has been computed.

3.5 Eliminating Stable Coalitions

Clamping units of the output layer allows to test candidate sets for explanations sequentially by, for example, first clamping the unit $e^\top$ in the output layer of $\mathcal{N}^{\circlearrowleft}_{\mathcal{P}_1}$, waiting until a stable state has been reached, withdrawing the external activation for $e^\top$ and clamping the unit $e^\bot$ thereafter.

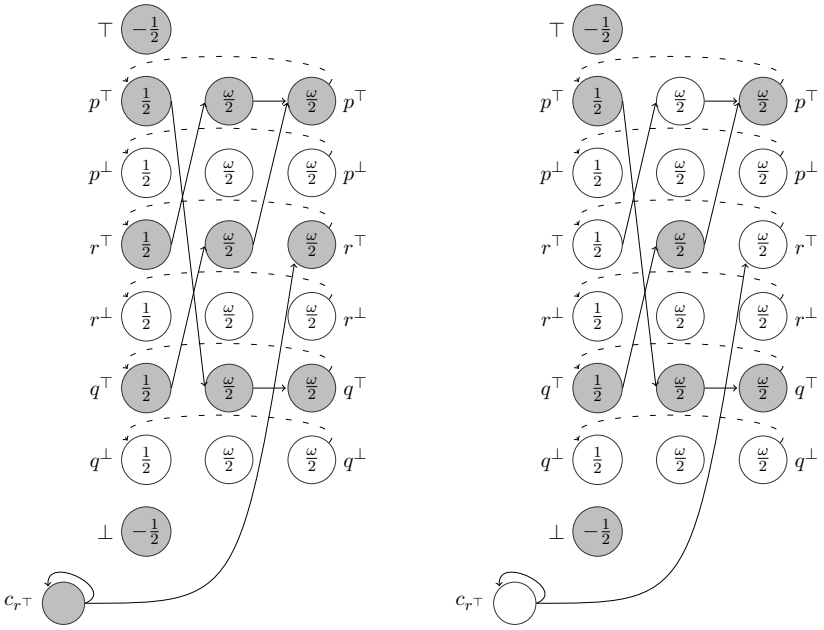

Figure 7: The stable state of the network $\mathcal{N}^{\circlearrowleft}_{\mathcal{P}_6}$ if the unit $r^\top$ in the output layer is externally clamped (left) and with the external activation withdrawn again (right). The connection from the unit $c_{r^\top}$ to the unit $r^\top$ has the weight ω.

However, there is the problem that a clamped unit may lead to a stable coalition which persists even if the external activation is withdrawn. As an example consider the program

$$\mathcal{P}_6 = \{p \leftarrow r,\ p \leftarrow q,\ q \leftarrow p\}$$

with $wc\mathcal{P}_6 = \langle \emptyset, \emptyset \rangle$. One should observe that $\mathcal{P}_6$ contains a *(positive) cycle*: p depends on q (because of $p \leftarrow q$) and q depends on p (because of $q \leftarrow p$). The recursive network $\mathcal{N}^{\circlearrowleft}_{\mathcal{P}_6}$ corresponding to $\mathcal{P}_6$ is shown in Figure 7. If the unit $r^\top$ in the output layer is externally clamped using unit $c_{r^\top}$, then the stable state of $\mathcal{N}^{\circlearrowleft}_{\mathcal{P}_6}$ represents the interpretation $\langle \{r, p, q\}, \emptyset \rangle$, which is the least Ł-model of $(\mathcal{P}_6 \cup \{r \leftarrow \top\})$.

If this external activation is withdrawn, then the stable state of the recurrent network represents the interpretation $\langle \{p, q\}, \emptyset \rangle$, which is not the least Ł-model of $\mathcal{P}_6$. The units representing p and q are forming a *stable coalition* which was triggered by the previous external activation of the unit representing $r^\top$ in the output layer. If we would continue by externally clamping the unit $r^\bot$ in the output layer, then the stable state of the network will represent the interpretation $\langle \{p, q\}, \{r\} \rangle$, which is not the least Ł-model of $(\mathcal{P}_6 \cup \{r \leftarrow \bot\})$.

Thus, in order to eliminate stable coalitions after some external activation has been withdrawn all units occurring in the input layer (including the units $\top$ and $\bot$) must be externally deactivated for three time steps. This can be achieved by (inhibitory) connections with weight -2. After three time steps, all units of the recurrent network will be passive and the external inhibition can be withdrawn. This is exemplified in Figure 8. At $t = 1$ (top left) all units in the input layer become passive as the sum of their weighted inputs is

less or equal than -1. At $t = 2$ (top right) all units in the hidden layer become passive as the sum of their weighted inputs is zero. Likewise, at $t = 3$ (bottom) all units in the output layer become passive. The inhibitory connections from the unit i to the units of the input layer have weight -2 and are depicted with a bullet at their tip.

Likewise, the unit $c_{r^\top}$ must be deactivated. This can be done by negatively modifying the self-excitatory connection from $c_{r^\top}$ to itself by the unit i (see also Figure 11).

3.6 Detecting Stable States

Let $\mathcal{N}_\mathcal{P}^\circlearrowleft$ be the recurrent neural network corresponding to the program $\mathcal{P}$. We need to determine whether $\mathcal{N}_\mathcal{P}^\circlearrowleft$ has reached a stable state. To this end we construct a neural network as follows. Let n be the number of units occurring in $\mathcal{N}_\mathcal{P}^\circlearrowleft$.

1. For each unit u occurring in $\mathcal{N}_\mathcal{P}^\circlearrowleft$ do the following.
   (a) Add a unit u' with threshold $\frac{1}{2}$ and connect u to u' with weight 1.
   (b) Connect u and u' with weight -1 to a new unit $v^\bot$ with threshold $-\frac{1}{2}$.
   (c) Connect u and u' with weight 1 to a new unit $v^\top$ with threshold $\frac{3}{2}$.
   (d) Connect $v^\bot$ and $v^\top$ with weight 1 to a new unit w with threshold $\frac{1}{2}$.
2. Connect the n w-units generated in the first step with weight 1 to a new unit s with threshold $n - \frac{1}{2}$.

For each unit u occurring in $\mathcal{N}_\mathcal{P}^\circlearrowleft$ the unit u' acts as a copy in the sense that at time $t + 1$ the unit u' is in the same state as u at time t. The unit $v^\bot$ becomes active if and only if both, u' and u, are passive at t and $t + 1$, respectively. In this case, the sum of the weighted input of $v^\bot$ is 0, which is above its threshold of $-\frac{1}{2}$. The unit $v^\top$ becomes active if and only if u' and u are active at time t and $t + 1$, respectively. In this case, the sum of the weighted input of $v^\top$ is 2, which is above its threshold of $\frac{3}{2}$. The unit w becomes active if and only if $v^\bot$ is active or $v^\top$ is active. In this case, the sum of the weighted input of w is 1, which is above its threshold of $\frac{1}{2}$; unit u has not changed its state between t and $t + 1$. Finally, the unit s becomes active if and only if no unit occurring in $\mathcal{N}_\mathcal{P}^\circlearrowleft$ has changed its state between t and $t + 1$. In this case, the sum of the weighted input received by the unit s is n and is above its threshold of $n - \frac{1}{2}$. The network is exemplified in Figure 9.

3.7 Checking Integrity Constraints

If the least Ł-model $\mathcal{M}_\mathcal{P}$ of a program $\mathcal{P}$ has been computed, i.e., if a stable state of the corresponding recurrent network $\mathcal{N}_\mathcal{P}^\circlearrowleft$ has been reached, then we may be forced to check whether the formulas occurring in a set $\mathcal{IC}$ of integrity constraints are satisfied under $\mathcal{M}_\mathcal{P}$. This can be done by a three-layered feed-forward network as follows. Let $\omega \in \mathbb{R}^+$.

1. The input layer of the new network is the output layer of the $\mathcal{N}_\mathcal{P}^\circlearrowleft$. The output layer consists of a single unit v with threshold $\frac{\omega}{2}$.
2. For each integrity constraint of the form $\mathsf{U} \leftarrow B_1 \wedge \ldots \wedge B_k$ occurring in $\mathcal{IC}$ add a binary threshold unit u with threshold $k\omega - \frac{\omega}{2}$ to the hidden layer. Connect u to v with weight w.

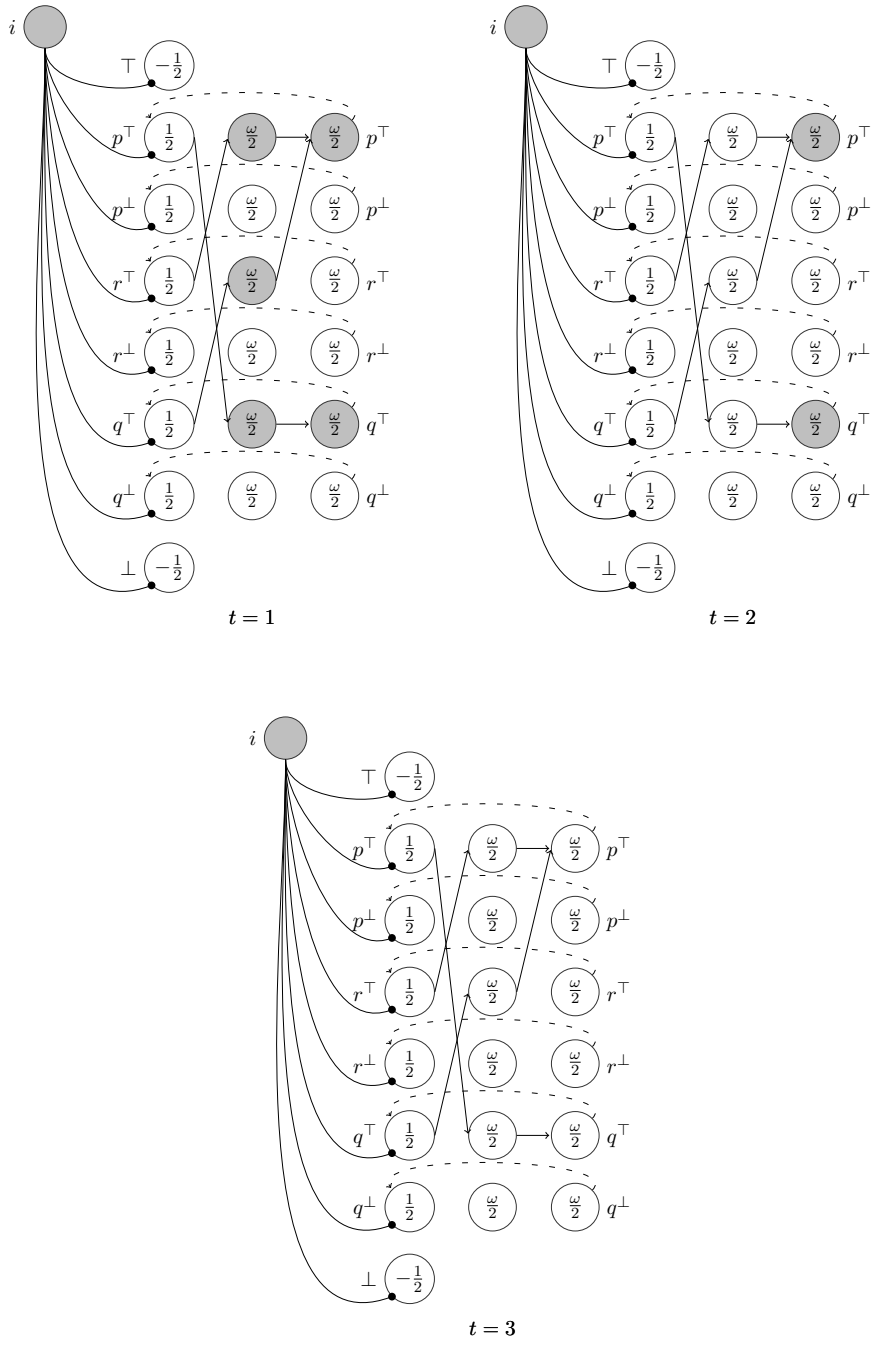

Figure 8: The network shown in Figure 7 after the external activation of the unit representing r in the output layer is withdrawn and the external inhibition of the units in the input layer is clamped for three time steps.

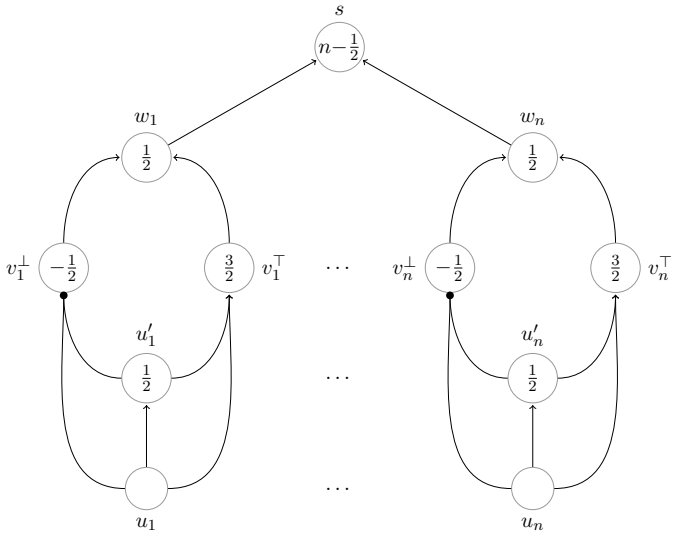

Figure 9: Detecting stable states in a network with n units.

3. Connect the units representing the literals B_i, $1 \leq i \leq k$, in the input layer to u with weight w.

The unit u in the hidden layer representing an integrity constraint $\mathsf{U} \leftarrow B_1 \wedge \ldots \wedge B_k$ will become active if and only if all units representing the literals B_i, $1 \leq i \leq k$, in the input layer are active. In this case, the sum of the weighted input received by u is $k\omega$, which is larger than its threshold of $k\omega - \frac{\omega}{2}$. The output unit v will become active if and only if one hidden layer unit becomes active. In this case, the sum of the weighted input of v is larger than ω, which is larger than its threshold of $\frac{\omega}{2}$. This is exemplified in Figure 10 using the program $\mathcal{P}_6$ and assuming the integrity constraint $\mathcal{IC} = \{\mathsf{U} \leftarrow q \wedge r\}$.

3.8 Checking Observations

A similar construction as in the previous subsection can be used to check whether all observations can be explained by the least Ł-model $\mathcal{M}_\mathcal{P}$ of a program $\mathcal{P}$. Let $\mathcal{O} = \{L_1, \ldots, L_k\}$ be the observation and $\omega \in \mathbb{R}^+$. Construct the following two-layered feed-forward network.

1. The input layer is the output layer of the recurrent neural network for $\mathcal{P}$. The output layer consists of a single unit o with threshold $k\omega - \frac{\omega}{2}$.

2. Connect the units representing L_i, $1 \leq i \leq k$, in the input layer to o with weight ω.

The unit o will become active if and only if all units representing the literals L_i, $1 \leq i \leq k$, in the input layer are active. In this case, the sum of the weighted input received by o is $k\omega$, which is larger than its threshold of $k\omega - \frac{\omega}{2}$. In other words, o will become active if and only if all observations can be explained. This can be checked as soon as the network corresponding to the program has reached a stable state. It takes one time step to compute the value of o. Figure 10 illustrates the case for the program $\mathcal{P}_6$ and the observation $\mathcal{O} = \{p, r\}$.

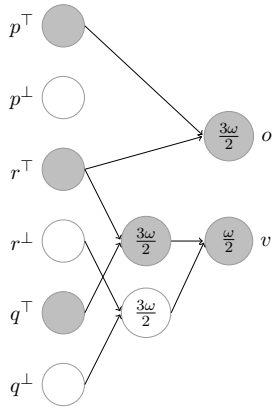

Figure 10: Checking the integrity constraint $\mathcal{IC} = \{\mathsf{U} \leftarrow q \wedge r\}$ and the observation $\mathcal{O} = \{p, r\}$ with respect to the program $\mathcal{P}_6$. All connections have the weight ω. If the units $p^\top$, $r^\top$, and $q^\top$ in the output layer of $\mathcal{N}^{\circlearrowleft}_{\mathcal{P}_6}$ are active, then $\mathcal{IC}$ is violated, but $\mathcal{O}$ is explained.

3.9 A Brief Summary

The semantic operator $\Phi_\mathcal{P}$ associated with a program $\mathcal{P}$ can be computed by a three-layer feed-forward network $\mathcal{N}_\mathcal{P}$, where the input and output layers represent interpretations. Turned into a recurrent network $\mathcal{N}^{\circlearrowleft}_\mathcal{P}$, it will converge to a stable state which represents the least Ł-model of $\mathcal{P}$. The network can be extended to clamp units in the output layer which correspond to possible explanations for a given observation. Once the network has settled down in a stable state, then it can be checked whether the observation can be explained and whether given integrity constraints are satisfied.

The external activation of units in the output layer can lead to stable coalitions which persist if the external activation is withdrawn. These coalitions can be eliminated by inhibiting the network for three time steps. All together, we obtain an extended recursive network $e\mathcal{N}^{\circlearrowleft}_\mathcal{P}$ which is depicted in Figure 11 for the program $\mathcal{P}_2$ and the observation $\mathcal{O} = \{\ell\}$.

4. A Connectionist Network for Skeptical Abduction

The approach presented in this section is inspired by Ray and d'Avila Garcez (2006) and d'Avila Garcez et al. (2007), but deviates in various aspects. Let $\mathcal{P}$ be a logic program, $\mathcal{N}^{\circlearrowleft}_\mathcal{P}$ be the corresponding recursive network, $\langle \mathcal{P}, \mathcal{A}_\mathcal{P}, \mathcal{IC}, \models_{wcs} \rangle$ be an abductive framework with respect to $\mathcal{P}$, and $\mathcal{O}$ be an observation.

As Ray and d'Avila Garcez (2006) and d'Avila Garcez et al. (2007) we will construct a network which sequentially generates all possible explanations for $\mathcal{O}$ and tests whether the least Ł-model of the program $\mathcal{P}$ together with a possible explanation satisfies the integrity constraint $\mathcal{IC}$ and explains the observation $\mathcal{O}$. However, whereas Ray and d'Avila Garcez and d'Avila Garcez et al. stop this process as soon as a (credulous) explanation is found we check all possible explanations and compute the skeptical consequences. Whereas Ray and

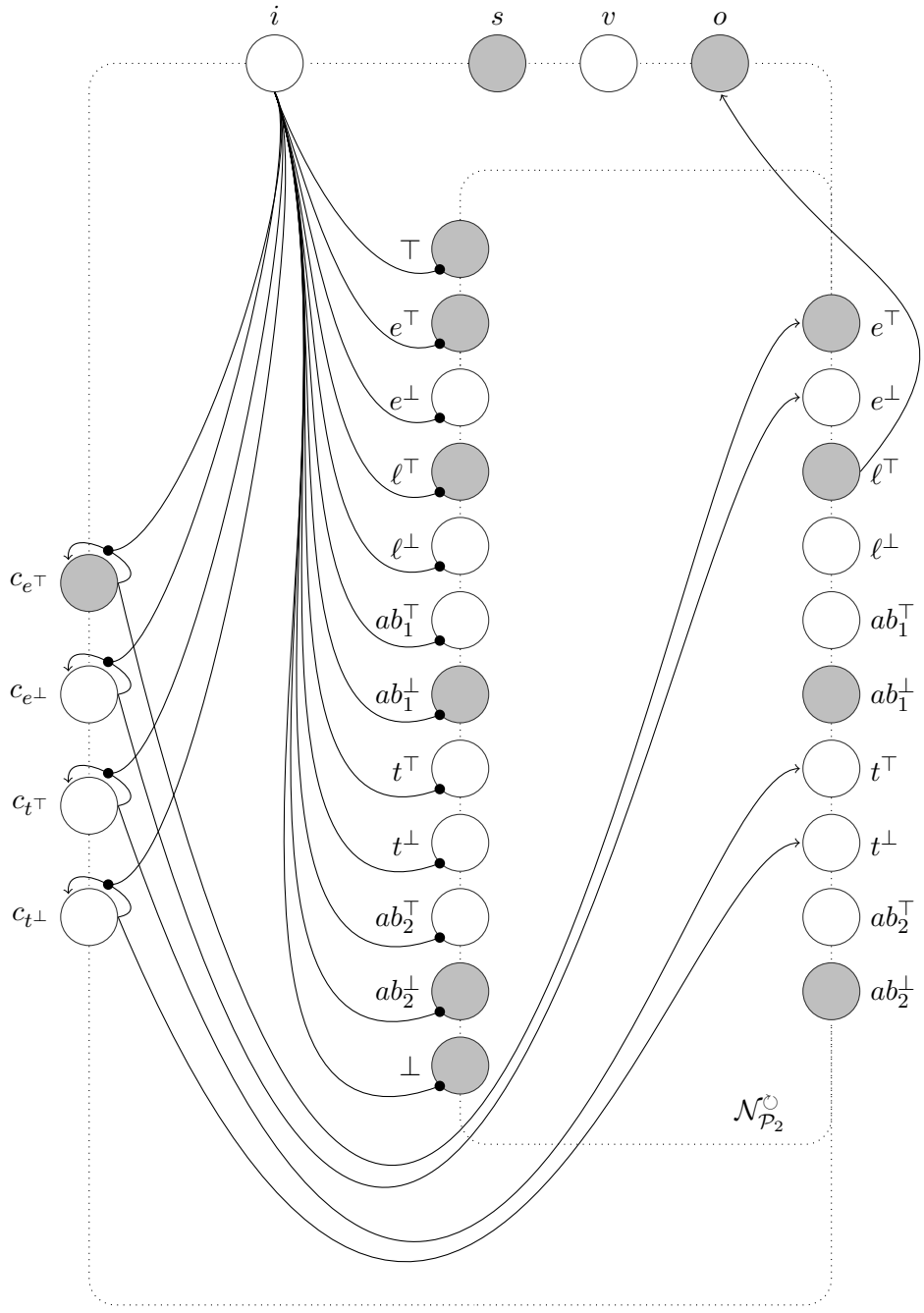

Figure 11: The extended recursive network $e\mathcal{N}^{\circlearrowleft}_{\mathcal{P}_2}$.

d'Avila Garcez and d'Avila Garcez et al. compute abduction in classical two-valued logic, we apply the three-valued Ł-logic and use $\models_{wcs}$ as the logical consequence relation.

There is another technical issue: Ray and d'Avila Garcez (2006) and d'Avila Garcez et al. (2007) have used a clock to ensure that the recursive network $\mathcal{N}_{\mathcal{P}}^{\circlearrowleft}$ has settled in a stable state. This can be done as long as the given program $\mathcal{P}$ is *definite*, i.e., if all bodies of the clauses occurring in $\mathcal{P}$ are conjunctions of atoms or $\top$ and do not contain occurrences of negated atoms and $\bot$. The corresponding semantic operator is known to have a least fixed point in classical two-valued logic, which can be obtained by iterating the operator starting with the empty interpretation (Apt & van Emden, 1982). As shown by Dowling and Gallier (1984) and Scutella (1990), the least fixed point can be computed in less or equal than $|atoms(\mathcal{P})|$ steps for a propositional and definite program $\mathcal{P}$, thus specifying a limit for the clock. However, if the program is not definite, then a least fixed point may not exist and, consequently, the recurrent network may not reach a stable state at all. Moreover, if the approach is extended to full first-order logic, then we may not be able to predict how long it may take to compute a least fixed point if it exists. To overcome these problems we use the construction developed in Subsection 3.6 to detect when $\mathcal{N}_{\mathcal{P}}^{\circlearrowleft}$ has settled down in a stable state. This construction is independent of the form of the program and the underlying logic.

4.1 Generating Possible and Non-complementary Explanations

If a given program $\mathcal{P}$ contains n undefined atoms, then the set $\mathcal{A}_{\mathcal{P}}$ of abducibles contains $2n$ elements and there are 3^n possible and non-complementary explanations. Reconsidering the program $\mathcal{P}_1 = \{\ell \leftarrow e \wedge \neg ab_1,\ ab_1 \leftarrow \bot\}$ we observe that only e is undefined. Hence, $n = 1$, $\mathcal{A}_{\mathcal{P}_1} = \{e \leftarrow \top,\ e \leftarrow \bot\}$ and we obtain the three possible and non-complementary explanations $\emptyset$, $\{e \leftarrow \top\}$, and $\{e \leftarrow \bot\}$. We can label these explanations by the binary numbers 00, 01, and 10, respectively, with the understanding that the first bit (from the right) represents the positive fact $e \leftarrow \top$ and the second bit (from the right) represents the negative assumption $e \leftarrow \bot$. Now, the task is to construct a neural network which sequentially generates the binary numbers 00, 01, and 10.

This task can be formally specified by a finite automaton with state output (Moore machine) as shown in Figure 12. This automaton has the set $\{0, 1\}$ of input symbols, the set $\{00, 01, 10\}$ of output symbols, the set $\{q_0, q_1, q_2\}$ of states with q_0 being the initial state, the transition function

	0	1
q_0	q_0	q_1
q_1	q_1	q_2
q_2	q_2	q_0

and the output function

q_0	q_1	q_2
00	01	10

.

Likewise, Figure 13 shows the corresponding automaton for the program $\mathcal{P}_2$ and the observation $\mathcal{O} = \{\ell\}$. This automaton outputs four binary numbers which represent from the right $e \leftarrow \top,\ e \leftarrow \bot,\ t \leftarrow \top$, and $t \leftarrow \bot$.

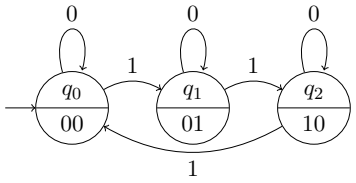

Figure 12: A finite automaton generating all possible and non-complementary explanations for $\mathcal{A}_{\mathcal{P}_1}$.

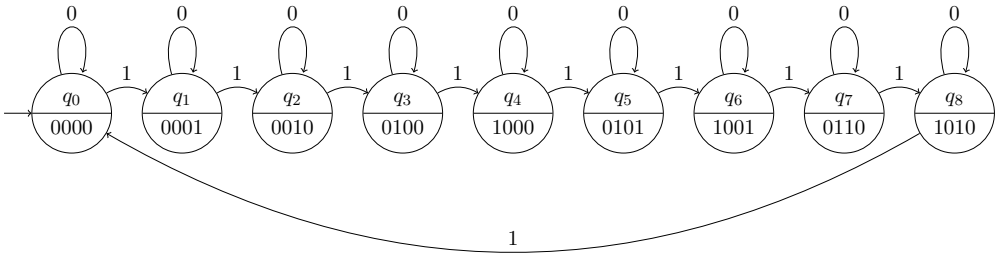

Figure 13: A finite automaton generating all possible and non-complementary explanations for $\mathcal{A}_{\mathcal{P}_2}$.

McCulloch and Pitts (1943) have shown in that each finite automaton with state output can be turned into a neural network of binary threshold units (sometimes called a *McCulloch-Pitts network*) receiving an input via its input units and producing an output via its output units. In particular, for the finite automaton shown in Figure 13 we can obtain a network $\mathcal{N}_{\mathcal{A}_{\mathcal{P}_2}}$ with an input unit t triggering the state transitions and four output units $e^\top$, $e^\bot$, $t^\top$, and $t^\bot$ (see Figure 14). Initially, the unit q_0 will be active. Because $\mathcal{E}_0 = \emptyset$, no output is generated by $\mathcal{N}_{\mathcal{A}_{\mathcal{P}_2}}$ (compare (1) on page 59). As soon as the unit t is externally activated, then the unit q_0 will become passive and the unit q_1 will become active. Moreover, the unit q_1 will activate the unit $e^\top$ representing the first possible, non-empty explanation $\mathcal{E}_1 = \{e \leftarrow \top\}$ (compare (2) on page 59). Upon further external activation of the unit t, this process will continue until the unit q_8 becomes active. The unit q_8 will activate the units $e^\bot$ and $t^\bot$ representing the last possible explanation $\mathcal{E}_8 = \{e \leftarrow \bot,\ t \leftarrow \bot\}$ (compare (7) on page 59). Another activation of the unit t will return the network into its initial state. Because the environment needs to know when all possible explanations have been generated, another output unit d (*done*) is foreseen which will also become active once the unit q_8 becomes passive. The unit d will remain active unless the unit t is externally activated again.

4.2 Computing Skeptical Conclusions

Let $\mathcal{P}$ be the given program, $m = atoms(\mathcal{P})$ be the number of atoms occurring in $\mathcal{P}$, and n the number of undefined atoms occurring in $\mathcal{P}$. The network $\mathcal{N}^{\circlearrowleft}_{\mathcal{P}}$ contains $2m$ output units,

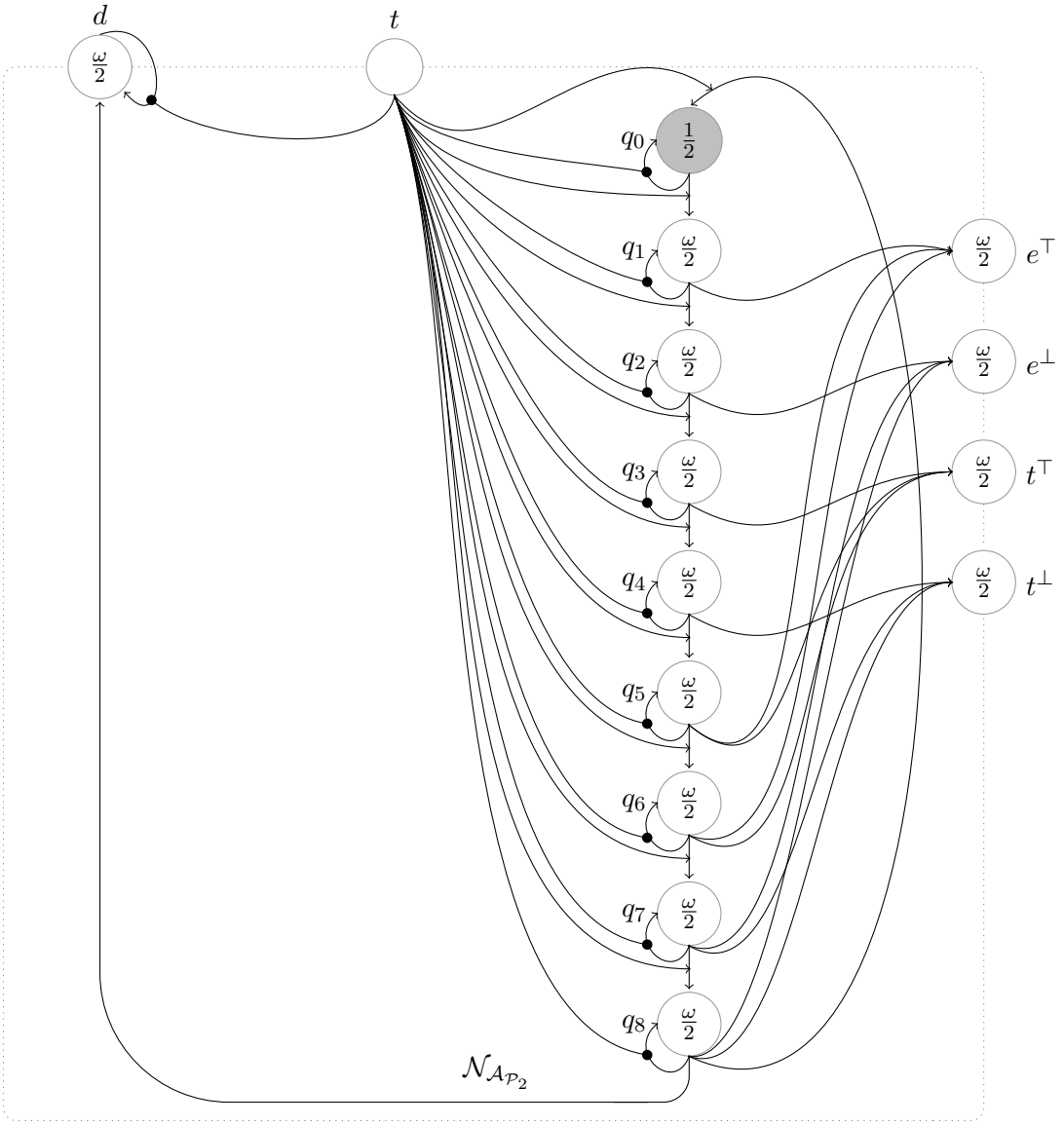

Figure 14: The McCulloch-Pitts network for the finite automaton shown in Figure 13. Unit q_0 is initially active; all other units are initially passive.

viz. for each $i \in [1, m]$ the units $i^\top$ and $i^\bot$. As discussed in Subsections 3.4 and 4.1, this network will be used to sequentially check whether the possible explanations $\mathcal{E}_j$ occurring in a sequence $(\mathcal{E}_j \mid 1 \leq j \leq 3^n)$ are explanations. For each $\mathcal{E}_j$ this is done by clamping the units of the output layer in $e\mathcal{N}_\mathcal{P}^\circlearrowleft$ corresponding to $\mathcal{E}_j$ and waiting until the network has settled down in a stable state. Such a stable state will eventually be reached and, hence, the unit s specified in Subsection 3.6 will become active. If the unit v specified in Subsection 3.7 is passive (no integrity constraint occurring in $\mathcal{IC}$ is violated) and the unit o specified in Subsection 3.8 is active (the observation $\mathcal{O}$ is explained), then $\mathcal{E}_j$ is indeed an explanation. In this case, the activation pattern of the output layer of $e\mathcal{N}_\mathcal{P}^\circlearrowleft$ (with the output units corresponding to $\mathcal{E}_j$ clamped) encodes the least Ł-model $\mathcal{M}_{\mathcal{P}\cup\mathcal{E}_j}$ as follows:

- $i^\top$ is active if and only if $\mathcal{M}_{\mathcal{P}\cup\mathcal{E}_j}(i) = \top$,
- $i^\bot$ is active if and only if $\mathcal{M}_{\mathcal{P}\cup\mathcal{E}_j}(i) = \bot$,
- $i^\top$ and $i^\bot$ are both passive if and only if $\mathcal{M}_{\mathcal{P}\cup\mathcal{E}_j}(i) = \mathsf{U}$,
- and it cannot be the case that both, $i^\top$ an $i^\bot$, are active.

A generated least Ł-model for a program $\mathcal{P}$ and an explanation $\mathcal{E}_j$ must be stored and compared to the least Ł-models of $\mathcal{P}$ and other explanations in order to compute skeptical conclusions. To this end we use a unit e (*extract*) which is defined by the rule

$$e \leftarrow s \wedge \neg v \wedge o$$

in classical two-valued logic. In other words, the unit e will become active if

- the recursive network $\mathcal{N}_\mathcal{P}^\circlearrowleft$ with the output units corresponding to $\mathcal{E}_j$ being externally clamped has reached a stable state (the unit s is active),
- no integrity constraints are violated (the unit v is passive), and
- the observation is explained (the unit o is active).

Unit e is part of the network depicted in Figure 16.

The skeptical conclusion are calculated by a two-layered network $\mathcal{N}_\mathcal{S}$. Its input and its output layer consists of $3m$ units: for each atom i occurring in $\mathcal{P}$ there are the units $i^\top$, $i^\bot$, and i^U. If the unit e is active, then the units in the input layer of $\mathcal{N}_\mathcal{S}$ are activated by the corresponding units in the output layer of $\mathcal{N}_\mathcal{P}^\circlearrowleft$:

$$\begin{array}{l}\{i^\top \leftarrow i^\top \wedge e \mid 1 \leq i \leq m\} \cup \\ \{i^\bot \leftarrow i^\bot \wedge e \mid 1 \leq i \leq m\} \cup \\ \{i^\mathsf{U} \leftarrow \neg i^\top \wedge \neg i^\bot \wedge e \mid 1 \leq i \leq m\},\end{array}$$

where the units $i^\top$, $i^\bot$, and i^U occurring on the left hand side of the rules are input units of $\mathcal{N}_\mathcal{S}$ and the units $i^\top$ and $i^\bot$ occurring on the right hand side of the rules are output units of $\mathcal{N}_\mathcal{P}^\circlearrowleft$. These rules can be easily implemented by a feed-forward network of logical threshold units following McCulloch and Pitts (1943), where we use the output of e as a

positive modifier for the connections between the output units of $\mathcal{N}_{\mathcal{P}}^{\circlearrowleft}$ and the input units of $\mathcal{N}_{\mathcal{S}}$. These connections are shown in Figure 16.

The input units of $\mathcal{N}_{\mathcal{S}}$ must be self-excitatory as a least Ł-model for the program and some explanation must persist until all explanations have been tested. If all explanations have been generated, then the activation pattern of the input units of $\mathcal{N}_{\mathcal{S}}$ shows the positive, negative, and undefined atoms, which can be credulously concluded from the given program $\mathcal{P}$ and the given observation $\mathcal{O}$. In order to generate the skeptical consequences we need to exclude those atoms whose interpretation varied under different least Ł-models, i.e., those atoms i for which more than one unit out of $\{i^{\top},\ i^{\bot},\ i^{\mathsf{U}}\}$ in the input layer of $\mathcal{N}_{\mathcal{S}}$ is active. This can be achieved by connecting each input unit of $\mathcal{N}_{\mathcal{S}}$ to the corresponding output unit of $\mathcal{N}_{\mathcal{S}}$ and, at the same time, negatively modifying competing connections between the input and the output layer. For example, the unit $i^{\top}$ in the input layer is connected to the unit $i^{\top}$ in the output layer and is negatively modifying the connections from the unit $i^{\bot}$ in the input layer to the unit $i^{\bot}$ in the output layer as well as from the unit i^{U} in the input layer to the unit i^{U} in the output layer.

Figure 15 shows the network $\mathcal{N}_{\mathcal{S}}$ for the program $\mathcal{P}_2$ and observation $\mathcal{O} = \{\ell\}$. Recall that the explanations $\mathcal{E}_1$, $\mathcal{E}_3$, $\mathcal{E}_5$, $\mathcal{E}_6$, and $\mathcal{E}_7$ explain $\mathcal{O}$ (see page 59). Comparing the least Ł-models of $\mathcal{P}_2$ together with these explanations we obtain the final pattern of the input units of $\mathcal{N}_{\mathcal{S}}$:

- $e^{\top}$ is active because of (2), (4), and (5),
- $e^{\bot}$ is active because of (6),
- e^{U} is active because of (3),
- $\ell^{\top}$ is active because it is the observation,
- $ab_1^{\bot}$ is active because of the assumption $ab_1 \leftarrow \bot \in \mathcal{P}_2$,
- $t^{\top}$ is active because of (3), (4), and (6),
- $t^{\bot}$ is active because of (5),
- t^{U} is active because of (2),
- $ab_2^{\bot}$ is active because of the assumption $ab_2 \leftarrow \bot \in \mathcal{P}_2$.

All other units occurring in the input layer of $\mathcal{N}_{\mathcal{S}}$ are passive.

Consequently, the units $\ell^{\top}$, $ab_1^{\bot}$, and $ab_2^{\bot}$ of the output layer of $\mathcal{N}_{\mathcal{A}}$ are the only active units in this layer once the network has reached a stable state. The literals ℓ, $\neg ab_1$ and $\neg ab_2$ are the skeptical consequences of $\mathcal{P}_2$ and $\mathcal{O}$.

4.3 Control

After the specification of the connectionist networks for the various sub-parts of a network for skeptical abduction, we need to combine the sub-parts. Given the abductive framework $\langle \mathcal{P}, \mathcal{A}_{\mathcal{P}}, \mathcal{IC}, \models_{wcs} \rangle$ and the observation $\mathcal{O}$ the final network (see Figure 16) is supposed to do the following:

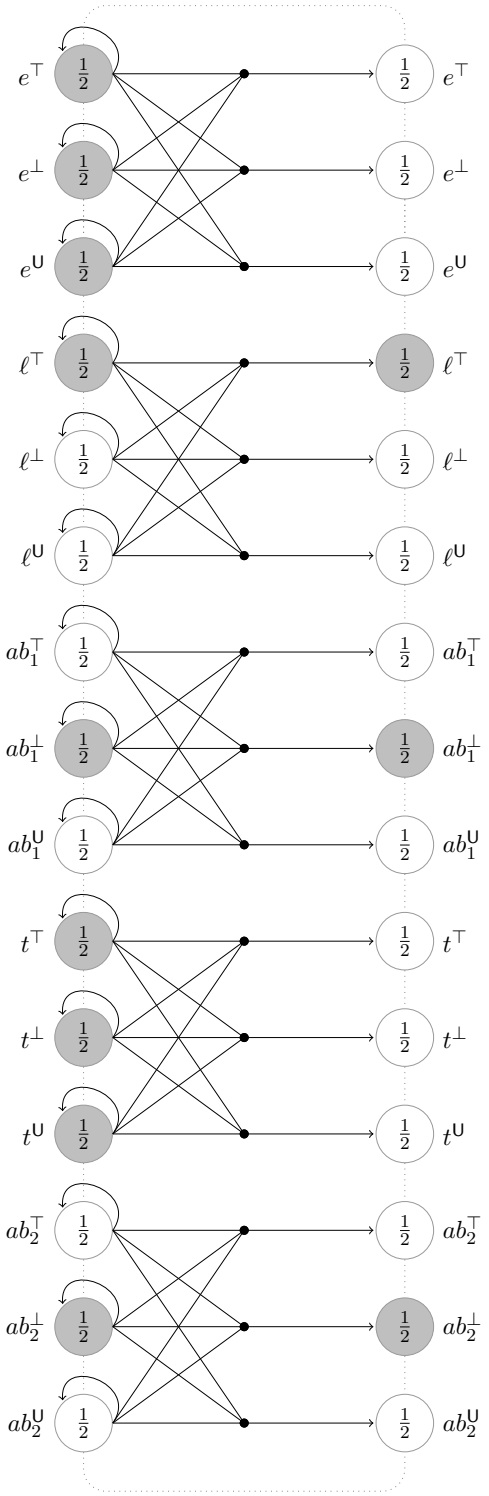

Figure 15: The network $\mathcal{N}_\mathcal{S}$ computing the skeptical consequences for the program $\mathcal{P}_2$ and observation $\mathcal{O} = \{\ell\}$. The activation pattern shows the stable state of the network.

1. While not all non-complementary subsets of $\mathcal{A}_\mathcal{P}$ have been tested do:
   (a) Delete stable states from $\mathcal{N}_\mathcal{P}^\circlearrowleft$.
   (b) Generate the next non-complementary subset $\mathcal{E}$ of $\mathcal{A}_\mathcal{P}$.
   (c) Clamp the output units of $\mathcal{N}_\mathcal{P}^\circlearrowleft$ which are corresponding to $\mathcal{E}$.
   (d) If $\mathcal{N}_\mathcal{P}^\circlearrowleft$ has reached a stable state, $\mathcal{IC}$ is not violated, and $\mathcal{O}$ can be explained, then add $\mathcal{E}$ to the set of explanations
2. Compute the skeptical consequences.

Step 1.(a) is implemented by externally activating the unit i of the network $e\mathcal{N}_\mathcal{P}^\circlearrowleft$ in three consecutive time steps. This is realized by three units *init*, i_1, and i_2 such that the unit *init* activates the unit i_1 and the unit i_1 activates the unit i_2 while all of them activate the unit i.

Step 1.(b) is implemented by externally activating for one time step the unit t of the network $\mathcal{N}_\mathcal{A}$. This is realized by adding a connection from the unit *init* to the unit t such that *init* activates the unit t. Once t becomes active, the next non-complementary $\mathcal{E} \subseteq \mathcal{A}_\mathcal{P}$ is generated as output pattern of $\mathcal{N}_\mathcal{A}$.

To implement step 1.(c) we introduce a new unit n (*next*), which is activated by the unit i_2 (stable coalitions occurring in $\mathcal{N}_\mathcal{P}^\circlearrowleft$ are removed) and positively modifies the connections from the output layer of $\mathcal{N}_\mathcal{A}$ and the corresponding input units of $e\mathcal{N}_\mathcal{P}^\circlearrowleft$.

Step 1.(d) is implemented by the unit e (top-right Figure 16) which acts as a positive modifier for the connections between the output layer of $e\mathcal{N}_\mathcal{P}^\circlearrowleft$ and the input layer of $\mathcal{N}_\mathcal{S}$.

The condition of the while-loop is tested with the help of the units s (of $e\mathcal{N}_\mathcal{P}^\circlearrowleft$), *init*, and d (of $\mathcal{N}_\mathcal{A}$). Initially, the unit *init* is externally activated. Thereafter, it is activated as soon as s becomes active and d is passive. In other words, as soon as the network $e\mathcal{N}_\mathcal{P}^\circlearrowleft$ has reached a stable state (unit s) and not all non-complementary subsets of $\mathcal{A}_\mathcal{P}$ have already been tested (unit d), unit *init* becomes active. With *init* active a new subset of $\mathcal{A}_\mathcal{P}$ is computed while stable coalitions occurring in $\mathcal{N}_\mathcal{P}^\circlearrowleft$ are removed. As the removal of stable coalitions from $\mathcal{N}_\mathcal{P}^\circlearrowleft$ takes time, we want to prevent that the unit s activates the unit *init* in subsequent time steps, the units *init*, i_1, i_2, and n negatively modify the connection from s to *init*.

Step 2 has already been explained in Section 4.2. It is implemented by the network $\mathcal{N}_\mathcal{S}$ and exemplified in Figure 15.

The final network for the program $\mathcal{P}_2$ and the observation $\mathcal{O} = \{\ell\}$ is shown in Figure 16.

5. Conclusion

Recall from the introduction, that in order to meet McCarthy's (1963) requirements for systems to reason about actions and causalities, we need to study humans and their behavior. It has been previously shown that skeptical abduction is required in order to adequately model a wide range of human reasoning tasks under the Weak Completion Semantics (WCS). Motivated by this observation, the main goal of this paper was to specify a plausible connectionist realization of WCS, that provides the representation of logic programs, the computation of least models, the interpretation under three-valued Łukasiewicz logic and the derivation of consequences under skeptical abduction.

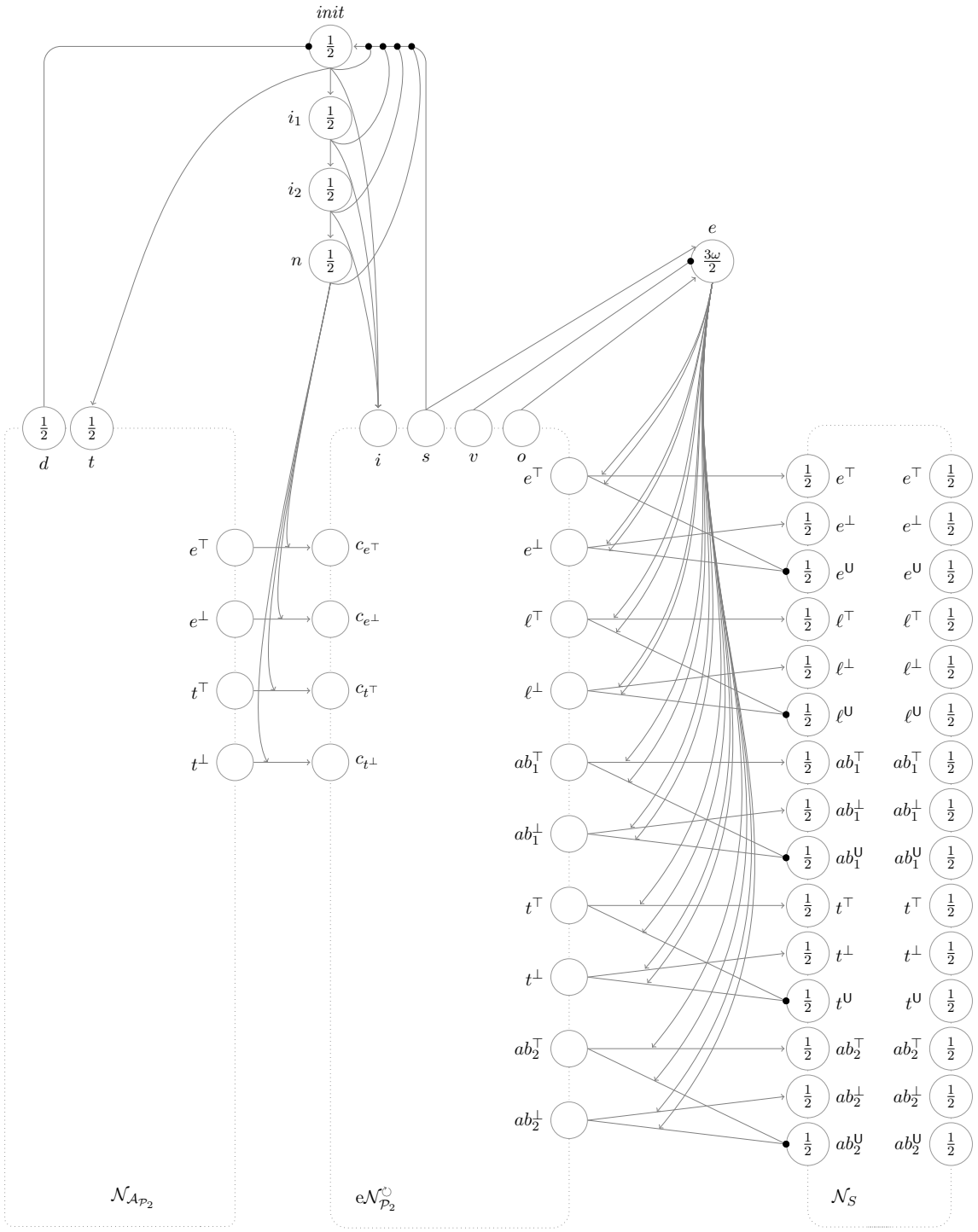

Figure 16: The final network for the program $\mathcal{P}_2$ and the observation $\mathcal{O} = \{\ell\}$. Networks $\mathcal{N}_{\mathcal{A}_{\mathcal{P}_2}}$, $e\mathcal{N}^{\heartsuit}_{\mathcal{P}_2}$ and $\mathcal{N}_{\mathcal{S}}$ are shown in Figure 14, 11 and 15, respectively.

As starting point we took the preliminary connectionist realization of the three-valued semantic operator considered under WCS (Hölldobler & Kencana Ramli, 2009b). This approach utilizes the CORE method where the semantic operator associated with a weakly completed program is mapped onto a feed-forward network. Additional recursive connections from the output to the input layer of this network allow for the iteration of the semantic operator until a least fixed point has been reached. This fixed point is the least model of the weakly completed program. Reasoning is performed with respect to this fixed point. Extending the work of Hölldobler and Kencana Ramli (2009b), skeptical abduction is added to the CORE method by sequentially generating all possible explanations, checking whether they are indeed explanations, and computing the abductive consequence of all explanations. We provide all details, a formal specification of the connectionist system, a proof of its soundness and several extensions: the possibility to clamp units representing atoms occurring in a possible extension, the check whether the network has reached a stable state, the elimination of stable coalitions, and the check whether observations are explained and integrity constraints are not violated. Observe that our approach combines different logics. The possible explanations for an abductive reasoning task are sequentially generated with the help of finite automata. The control, the computation of the credulous consequences, and the checks whether observations are explained and integrity constraints are not violated are done in classical two-valued logic. The generation of models for a particular human reasoning tasks are done in a non-monotonic and three-valued logic. A preliminary version of skeptical abduction – still under classical two-valued logic – was published by Dietz, Hölldobler, and Palacios Medinacelli (2015) and Palacios Medinacelli (2015).

The extension to skeptical abduction in this paper was inspired by Ray and d'Avila Garcez (2006) and d'Avila Garcez et al. (2007) who have developed a connectionist realization of credulous abduction in classical two-valued logic for definite logic programs. We share the idea to sequentially generate all possible extensions and to check whether a possible extension is indeed an extension by clamping corresponding units in the network realizing the semantic operator. But we deviate and extend the approach in various directions: Our programs are normal logic programs in that clauses may have negated atoms in their bodies, we switch to three-valued Łukasiewicz logic, we allow the network to settle down to a stable state without assuming a maximum time limit, we consider all possible explanations, and combine the consequences of all explanations.

Stenning and van Lambalgen have also introduced a connectionst realization to compute the least fixed point of their semantic operator (Stenning & van Lambalgen, 2008). Their networks are essentially identical to the networks generated by the CORE method for three-valued logics except that the units $A^\top$ and $A^\bot$ for an atom A in the input and output layer are connected by inhibitory connections to prevent both units to become active at the same time. As this can never happen in our approach, there is no need for such inhibitory connections. Moreover, their networks do not implement abduction and, hence, cannot handle examples like the ones discussed in this paper.

We have developed the networks using logical threshold units. But it is well-known (e.g. Bader, 2009, d'Avila Garcez et al., 1997) that these units can be replaced by sigmoidal ones such that the cores of the networks can be trained by back-propagation. During training some of the connections might change and rule extraction techniques must be applied to extract the trained programs.

But much remains to be done. While we have restricted possible explanations to non-complementary sets of abducibles, abduction often restricts explanations to minimal ones. Detecting minimality is not a trivial task. Palacios Medinacelli (2016) provided a formal specification that produces all possible explanations in a specific order such that minimal explanations can be detected and all non-minimal possible explanations can be discarded. More recently, Rocha (2017) added minimality to the approach presented in this paper. Moreover, she replaced the McCulloch-Pitts networks used to generate all possible explanations in a fixed and pre-defined sequence by Elman (1989, 1990) networks and showed that these networks can be trained to generate all possible explanations in an arbitrary order. This is a prerequisite for the next step: We do not believe that humans test all possible explanations in a systematic way. It appears to us that in particular reasoning episodes some possible explanations are systematically tested whereas others are not considered at all. We believe that there is a kind of attention formalism which identifies those possible explanations, which are really tested. This will lead to a kind of bounded skeptical abduction. But it remains to set up experiments that test this hypothesis and, if our hypothesis is supported, then we need to identify the mechanism which defines the bound and build it into our networks.

References

Apt, K., & van Emden, M. (1982). Contributions to the theory of logic programming. *Journal of the ACM*, *29*, 841–862.

Bader, S. (2009). *Neural-Symbolic Integration*. Ph.D. thesis, Technische Universität Dresden, Faculty of Computer Science.

Bader, S., Hitzler, P., & Hölldobler, S. (2008). Connectionist model generation: A first-order approach. *Neurocomputing*, *71*, 2420–2432.

Bader, S., & Hölldobler, S. (2006). The Core method: Connectionist model generation. In Kollias, S., Stafylopatis, A., Duch, W., & Ojaet, E. (Eds.), *Proceedings of the 16th International Conference on Artificial Neural Networks*, Vol. 4132 of *Lecture Notes in Computer Science*, pp. 1–13. Springer-Verlag.

Byrne, R. (1989). Suppressing valid inferences with conditionals. *Cognition*, *31*, 61–83.

Clark, K. (1978). Negation as failure. In Gallaire, H., & Minker, J. (Eds.), *Logic and Databases*, pp. 293–322. Plenum, New York.

d'Avila Garcez, A., Gabbay, D., Ray, O., & Woods, J. (2007). Abductive reasoning in neural-symbolic learning systems. *Topoi: An International Review of Philosophy*, *26*, 37–49.

d'Avila Garcez, A., Lamb, L. C., & Gabbay, D. (2002). A connectionist inductive learning system for modal logic programming. In *Proceedings of the IEEE International Conference on Neural Information Processing*, Singapore.

d'Avila Garcez, A., Lamb, L., & Gabbay, D. (2003). Neural-symbolic intuitionistic reasoning. In *Design and Application of Hybrid Intelligent Systems*, pp. 399–408, IOS Press.

d'Avila Garcez, A., Lamb, L., & Gabbay, D. (2009). *Neural-Symbolic Cognitive Reasoning*. Springer Verlag.

d'Avila Garcez, A., Zaverucha, G., & de Carvalho, L. (1997). Logic programming and inductive learning in artificial neural networks. In Herrmann, C., Reine, F., & Strohmaier, A. (Eds.), *Knowledge Representation in Neural Networks*, pp. 33–46, Berlin. Logos Verlag.

Dietz, E.-A., Hölldobler, S., & Palacios Medinacelli, L. (2015). A connectionist network for skeptical abduction. In Besold, T. R., Lamb, L. C., Icard, T., & Mikkulainen, R. (Eds.), *Proceedings of the Tenth International Workshop on Neural-Symbolic Learning and Reasoning*.

Dowling, W. F., & Gallier, J. H. (1984). Linear-time algorithms for testing the satisfiability of propositional Horn formulae. *Journal of Logic Programming*, *1*(3), 267–284.

Elman, J. L. (1989). Structured representations and connectionist models. In *Proceedings of the Annual Conference of the Cognitive Science Society*, pp. 17–25.

Elman, J. L. (1990). Finding structure in time. *Cognitive Science*, *14*, 179–211.

Feldman, J. A., & Ballard, D. H. (1982). Connectionist models and their properties. *Cognitive Science*, *6*(3), 205–254.

Fitting, M. (1985). A Kripke–Kleene semantics for logic programs. *Journal of Logic Programming*, *2*(4), 295–312.

Funahashi, K.-I. (1989). On the approximate realization of continuous mappings by neural networks. *Neural Networks*, *2*, 183–192.

Hitzler, P., Hölldobler, S., & Seda, A. (2004). Logic programs and connectionist networks. *Journal of Applied Logic*, *2*(3), 245–272.

Hölldobler, S. (2009). *Logik und Logikprogrammierung 1: Grundlagen*. Kolleg Synchron. Synchron.

Hölldobler, S. (2015). Weak completion semantics and its applications in human reasoning. In Furbach, U., & Schon, C. (Eds.), *Bridging 2015 – Bridging the Gap between Human and Automated Reasoning*, Vol. 1412 of *CEUR Workshop Proceedings*, pp. 2–16. CEUR-WS.org. `http://ceur-ws.org/Vol-1412/`.

Hölldobler, S., & Kalinke, Y. (1994). Towards a new massively parallel computational model for logic programming. In *Proceedings of the ECAI94 Workshop on Combining Symbolic and Connectionist Processing*, pp. 68–77. ECCAI.

Hölldobler, S., Kalinke, Y., & Störr, H.-P. (1999). Approximating the semantics of logic programs by recurrent neural networks. *Applied Intelligence*, *11*, 45–59.

Hölldobler, S., Kalinke, Y., & Wunderlich, J. (2000). A recursive neural network for reflexive reasoning. In Wermter, S., & Sun, R. (Eds.), *Hybrid Neural Symbolic Integration*, No. 1778 in Lecture Notes in Artificial Intelligence, pp. 46–62. Springer-Verlag.

Hölldobler, S., & Kencana Ramli, C. D. P. (2009a). Logic programs under three-valued Łukasiewicz's semantics. In Hill, P. M., & Warren, D. S. (Eds.), *Logic Programming*, Vol. 5649 of *Lecture Notes in Computer Science*, pp. 464–478. Springer-Verlag Berlin Heidelberg.

Hölldobler, S., & Kencana Ramli, C. D. P. (2009b). Logics and networks for human reasoning. In Alippi, C., Polycarpou, M. M., Panayiotou, C. G., & Ellinasetal, G. (Eds.), *19th International Conference on Artificial Neural Networks*, Vol. 5769 of *Lecture Notes in Computer Science*, pp. 85–94. Springer-Verlag Berlin Heidelberg.

Hornik, K., Stinchcombe, M., & White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks*, *2*, 359–366.

Johnson-Laird, P. N., & Byrne, R. M. J. (1991). *Deduction*. Lawrence Erlbaum Associates, Hove and London (UK).

Kalinke, Y. (1994). Ein massiv paralleles Berechnungsmodell für normale logische Programme. Master's thesis, Fakultät Informatik, TU Dresden. (in German).

Kencana Ramli, C. D. P. (2009). Logic programs and three-valued consequence operators. Master's thesis, International Center for Computational Logic, TU Dresden.

Khemlani, S., & Johnson-Laird, P. N. (2012). Theories of the syllogism: A meta-analysis. *Psychological Bulletin*, *138*(3), 427–457.

Kleene, S. (1952). *Introduction to Metamathematics*. North-Holland.

Lloyd, J. W. (1984). *Foundations of Logic Programming*. Springer, New York, USA.

Lourêdo Rocha, I. (2017). Bounded sceptical reasoning. Master's thesis, International Center for Computational Logic, TU Dresden.

Łukasiewicz, J. (1920). O logice trójwartościowej. *Ruch Filozoficzny*, *5*, 169–171. English translation: On Three-Valued Logic. In: *Jan Łukasiewicz Selected Works*. (L. Borkowski, ed.), North Holland, 87-88, 1990.

McCarthy, J. (1963). Situations and actions and causal laws. Stanford Artificial Intelligence Project: Memo 2.

McCulloch, W. S., & Pitts, W. (1943). A logical calculus and the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, *5*, 115–133.

Palacios Medinacelli, L. (2015). A connectionist model for skeptical abduction. Project Work, International Center for Computational Logic, TU Dresden.

Palacios Medinacelli, L. (2016). Skeptical abduction: A connectionist network. Master's thesis, International Center for Computational Logic, TU Dresden.

Ray, O., & d'Avila Garcez, A. (2006). Towards the integration of abduction and induction in artificial neural networks. In *Proceedings of the ECAI06 Workshop on Neural-Symbolic Learning and Reasoning*, pp. 41–46.

Ruiz, C., & Minker, J. (1995). Computing stable and partial stable models of extended disjunctive logic programs. In Dix, J., Pereira, L. M., & Przymusiski, T. C. (Eds.), *Non-Monotonic Extensions of Logic Programming*, Vol. 927 of *Lecture Notes in Artificial Intelligence*, pp. 205–229. Springer.

Rumelhart, D. E., Hinton, G. E., & McClelland, J. L. (1986). A general framework for parallel distributed processing. In *Parallel Distributed Processing*. MIT Press.

Scutella, M. G. (1990). A note on Dowling and Gallier's top-down algorithm for propositional Horn satisfiability. *Journal of Logic Programming*, *8*, 265–273.

Seda, A., & Lane, M. (2004). Some aspects of the integration of connectionist and logic-based systems. In *Proceedings of the Third International Conference on Information*, pp. 297–300, International Information Institute, Tokyo, Japan.

Stenning, K., & van Lambalgen, M. (2008). *Human Reasoning and Cognitive Science*. MIT Press.

Wason, P. C. (1968). Reasoning about a rule. *The Quarterly Journal of Experimental Psychology*, *20*, 273–281.