Possibilistic Nested Logic Programs

Juan Carlos Nieves and Helena Lindgren

Department of Computing Science
Umeå University
SE-901 87, Umeå, Sweden
`jcnieves,helena@cs.umu.se`

Abstract

We introduce the class of possibilistic nested logic programs. These possibilistic logic programs allow us to use nested expressions in the bodies and the heads of their rules. By considering a possibilistic nested logic program as a possibilistic theory, a construction of a possibilistic logic programing semantics based on answer sets for nested logic programs and the proof theory of possibilistic logic is defined. We show that this new semantics for possibilistic logic programs is computable by means of transforming possibilistic nested logic programs into possibilistic disjunctive logic programs. The expressiveness of the possibilistic nested logic programs is illustrated by scenarios from the medical domain. In particular, we exemplify how possibilistic nested logic programs are expressive enough for capturing medical guidelines which are pervaded of vagueness and qualitative information.

1998 ACM Subject Classification I.2.3 Deduction and Theorem Proving

Keywords and phrases Answer Set Programming, Uncertain Information, Possibilistic Reasoning

Digital Object Identifier 10.4230/LIPIcs.ICLP.2012.267

1 Introduction

In the literature, one can find different approaches for encoding qualitative information [12, 18, 20]. A common strategy for capturing qualitative information is by using non-numerical values. Possibilistic reasoning has shown to be a suitable approach for dealing with qualitative reasoning [18]. In particular, this feature is based on the fact that the possibilistic values of a possibilistic knowledge base can be non-numerical values which capture the uncertainty of a knowledge base.

In the context of possibilistic logic programming, there are few proposals which deal with non-numerical values which are not totally ordered [14]. However, the expressiveness of the approach presented in [14] is restricted to disjunctive logic programs. Indeed most of the logic programming approaches which deal with uncertain information make syntactic restrictions to their specification languages. By not having syntactic restriction in a symbolic specification, one can provide a transparent method to capture real data domains. For instance, there are different ways to interpret a medical guideline for diagnosis (we will illustrate this in the body of the paper). The presence of more than one disease in an individual (comorbidity) is common in older people, and some guidelines have expressions supporting both potential comorbidity and differential diagnosis. For example, the most frequently used guideline for mental diseases uses a multiple axis system between certain guidelines for expressing comorbidity [2]. Still, additional diagnostic criteria are needed to assess a potential dementia disease, which use a different way to express the ambiguity built into diagnosis of neurological and mental diseases. The uncertainty is reflected in the vocabulary

Technical Communications of the 28th International Conference on Logic Programming (ICLP'12).
Editors: A. Dovier and V. Santos Costa; pp. 267–276

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

used in these guidelines (e.g., possible, probable, unlikely, supportive, etc.). The meaning of such expressions represents different and sometimes overlapping ranges of possibilities, which, consequently, cannot be *totally ordered* when are formalized. However, there are practical reasons for reusing the vocabulary in the guidelines for expressing the knowledge in formal reasoning. Firstly, to provide to a clinician explanations of the reasoning and to mirror the uncertainty of an assessment in the available evidence-based medical knowledge, and secondly, to allow an expert physician to validate a knowledge base which handles comorbidity. An example of a possibilistic rule, which captures both uncertainty, ambiguity and a potential multi-diagnosis, is the following: ***possible***: $DLB \wedge AD \leftarrow visHall \wedge slow \wedge prog \wedge epiMem$. (It is possible that both Alzheimer's disease and Lewy Body dementia are present based on the observed symptoms). Another example illuminates how negation as failure can be utilized: ***probable***: $VaD \leftarrow fn \wedge radVasc \wedge\ not\ (AD \vee DLB)$ (vascular dementia is probable present considering the observations and since we do not have reasons to believe that Alzheimer's disease or Lewy Body dementia are present).

Against this background, we extended the results presented in [14] and [10] by introducing the class of *possibilistic nested logic programs*. These possibilistic logic programs allow us to use nested expressions in the bodies and the heads of their rules. Given that possibilistic logic is *axiomatizable* in the necessity-value case [6], we define the semantics of the possibilistic nested logic programs by considering the *proof theory* of possibilistic logic. In particular, by considering a possibilistic nested logic program as a possibilistic theory, a construction of a possibilistic logic programing semantics based on answer sets for nested logic programs [10] and the proof theory of possibilistic logic [6] is defined. It is worth mentioning that the answer set semantics inference can also be characterized as a *logic inference* in terms of the proof theory of intuitionistic logic and intermediate logics [17, 16].

We also show that the new possibilistic semantics generalizes the previous possibilistic semantics introduced in [13, 14]. In order to define a general method for computing the possibilistic answer sets of a possibilistic nested program, the idea of equivalence between possibilistic programs is explored.

The rest of the paper is divided as follows: In the following section, some basic concepts of nested logic programs and possibilistic logic are introduced. After this, the syntaxis and semantics of the possibilistic nested logic programs are introduced. In this section, some properties of the possibilistic nested logic semantics are identified (by lack of space, the formal proofs are omitted). In the last section, an outline of our conclusions and future work is presented.

2 Background

In this section, we introduce some basic concepts of Nested Logic Programs [10] and Possibilistic Logic [6]. We assume that the reader is familiarized with basic concepts in classical logic and logic programming semantics, *e.g.* interpretations, models, *etc.* A good introductory treatment of these concepts can be found in [3].

2.1 Nested Logic Programs

The considered language consists of: (i) an enumerable set $\mathcal{A}$ of elements called *atoms* (denoted by a, b, c, ...), (ii) *logic connectives* $\wedge$, $\vee$, $\neg$, *not*, $\bot$, $\top$ in which $\{\wedge, \vee\}$, $\{not, \neg\}$, $\{\top,\bot\}$ are 2-place, 1-place and 0-place connectives respectively and (iii) *auxiliary symbols* "(", ")", ".". We refer to a *literal* as an atom a or an extended atom $\neg a$. We denote by $\mathcal{L}$ the set of literals built using elements in $\mathcal{A}$.

Literals, $\bot$ and $\top$ are considered *elementary formulas*, while $\{\vee, \wedge, not\}$ *formulas* (denoted as A, B, C, ...) are constructed from elementary formulas using the connectives $\{\vee, \wedge, not\}$ arbitrarily nested (strong negation $\neg$ is allowed to appear only in front of atoms). As probably noted, we are considering two types of negations in this paper: strong negation $\neg$ (as it called by the Answer Set Programming community [3]) and negation as failure *not*.

Given a finite set of literals $\mathcal{L}$, a *nested rule* is an expression of the form $H \leftarrow B$, where H and B are either an elementary formula or a $\{\vee, \wedge, not\}$ formulas (known as the *head* and the *body* respectively). Some particular cases are *facts*, of the form $H \leftarrow \top$ (written as H), and *constraints*, $\bot \leftarrow B$ (written as $\leftarrow B$). If no occurrences of *not* appear in a rule, then the rule is called a *definite nested rule*.

A *nested logic program* P is a finite set of nested rules. If the program does not contain *not*, then the program is called a *definite nested program*.

The semantics for nested programs was introduced in [10]. Like the classic answer set semantics [7], the semantics for nested logic programs is defined in two steps: first for definite nested logic programs and after for general nested logic programs (programs which contain negation as failure).

▶ **Definition 1.** [10] Let M be a set of literals. M satisfies a definite nested formula A (denoted by $M \models A$), recursively as follows:

- for elementary A, $M \models A$ if $A \in M$ or $A = \top$
- $M \models A \wedge B$ if $M \models A$ and $M \models B$
- $M \models A \vee B$ if $M \models A$ or $M \models B$

▶ **Definition 2.** [10] Let P be a definite nested logic program. A set of literals M is closed under P if, $\forall r \in P$ such that $r = H \leftarrow B$, $M \models H$ whenever $M \models B$.

▶ **Definition 3.** [10] Let M be a set of literals and P a definite nested logic program. M is called an answer set for P if M is minimal among the consistent sets of literals closed under P.

In order to manage the negation as failure in nested logic programs, a syntactic reduction for nested logic programs was defined.

▶ **Definition 4.** [10] The reduction of a nested formula with respect to a set of literals M is recursively defined as follows:

- for elementary A, $A^M = A$
- $(A \wedge B)^M = A^M \wedge B^M$
- $(A \vee B)^M = A^M \vee B^M$
- $(not\ A)^M = \begin{cases} \bot, & \text{if } M \models A^M \\ \top, & \text{otherwise} \end{cases}$
- $(H \leftarrow B)^M = H^M \leftarrow B^M$

▶ **Definition 5.** [10] The reduct of a nested logic program P^M with respect to a set of literals M is defined as follows:

- $P^M = \{(H \leftarrow B)^M \mid H \leftarrow B \in P\}$

Please observe that P^M is a definite nested logic program. Hence, the following definition follows from the answer set definition.

▶ **Definition 6.** [10] Let P be a nested logic program and M be a set of literals. M is an answer set of P if it is an answer set of P^M.

Table 1 Examples of possibilistic rules captured by the syntax of possibilistic nested programs.

Syntax	Rule Type
$\alpha : a \wedge not\ b \leftarrow p \wedge not\ (\neg q \vee r).$	*possibilistic nested rule*
$\alpha : a \vee b \leftarrow c \wedge not\ \neg e.$	*possibilistic disjunctive rule* [14]
$\alpha : a \leftarrow c \wedge not\ d.$	*possibilistic normal rule* [13]

3 Possibilistic Nested Logic Programs

In this section, the general syntax and semantics for possibilistic nested logic programs will be presented. We will show that the semantics of the possibilistic nested logic programs generalizes the logic programming semantics of both the nested logic programs and the possibilistic disjunctive logic programs (the particular case of possibilistic normal logic programs is also considered). In order to define a process for computing the possibilistic answer sets of a possibilistic nested logic program some transformations between possibilistic programs are formalized.

The syntax of the possibilistic nested logic programs is based on the standard syntax of nested logic programs.

3.1 Syntax

We start by defining some concepts for managing the possibilistic values of a possibilistic knowledge base. We want to point out that in the whole document only finite lattices are considered.

A *possibilistic atom* is a pair $p = (a, q) \in \mathcal{A} \times \mathcal{Q}$, in which $\mathcal{A}$ is a finite set of atoms and $(\mathcal{Q}, \leq)$ is a lattice. The projection $*$ to a possibilistic atom p is defined as follows: $p^* = a$. Also given a set of possibilistic atoms S, $*$ over S is defined as follows: $S^* = \{p^* | p \in S\}$.

Let $(\mathcal{Q}, \leq)$ be a lattice. A possibilistic nested rule r is of the form:

$$\alpha : A \leftarrow B$$

in which $\alpha \in \mathcal{Q}$ and $A \leftarrow B$ is a nested rule. The projection $*$ for a possibilistic nested rule is $r^* = A \leftarrow B$. On the other hand, the projection n for a possibilistic nested rule is $n(r) = \alpha$. This projection denotes the degree of necessity captured by the certainty level of the information described by r. A possibilistic nested constraint c is of the form:

$$\top_{\mathcal{Q}} : \ \leftarrow B$$

in which $\top_{\mathcal{Q}}$ is the top of the lattice $(\mathcal{Q}, \leq)$ and $\leftarrow B$ is a nested constraint as defined in the background section. The projection $*$ for a possibilistic nested constraint c is: $c^* = \ \leftarrow B$.

A possibilistic nested program P is a tuple of the form $\langle (\mathcal{Q}, \leq), N \rangle$, in which N is a finite set of possibilistic nested rules and possibilistic nested constraints. The generalization of $*$ over P is as follows: $P^* = \{r^* | r \in N\}$. If N^* is a set of nested definite rules, P is called a possibilistic nested definite logic program. Different formula combinations lead to different logic rules as shown in Table 1.

We illustrate a possibilistic nested program with an example from the dementia domain (simplified due to space reasons). A summary of the clinical guidelines which are used in

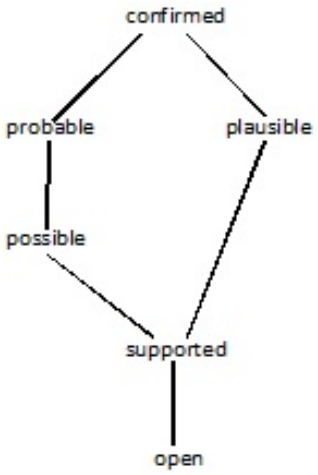

Figure 1 Graph representation of a lattice.

the dementia example given here can be found in [15] and includes [2]. We use the following abbreviations:

$$
\begin{aligned}
AD &= \textit{Alzheimer's disease}\\
DLB &= \textit{Lewy body type of dementia}\\
VaD &= \textit{Vascular dementia}\\
epiMem &= \textit{Episodic memory dysfunction}\\
fluctCog &= \textit{Fluctuating cognition}\\
fn &= \textit{Focal neurological signs}\\
prog &= \textit{Progressive course}\\
radVasc &= \textit{Radiology exam shows vascular signs}\\
slow &= \textit{Slow, gradual onset}\\
extraPyr &= \textit{Extrapyramidal symptoms}\\
visHall &= \textit{Visual hallucinations}
\end{aligned}
$$

We extract the following labels describing different levels of uncertainty of assessments from the clinical guidelines: $\mathcal{Q} := \{confirmed, probable, possible, plausible, supported, open\}$. To describe their relationships, let $<$ be a partial order such that the following set of relations holds: $\{confirmed > probable, probable > possible, confirmed > plausible, plausible > supported, possible > supported, supported > open\}$, see Figure 1. Given $x, y \in \mathcal{Q}$, the relation $x > y$ means that y is less certain than x.

▶ **Example 7.** The following clauses are included in our possibilistic nested logic program:

1. **probable**: $VaD \leftarrow fn \wedge radVasc \wedge\ not\ (AD \vee DLB)$
2. **probable**: $DLB \leftarrow extraPyr \wedge visHall \wedge not\ fn$
3. **probable**: $DLB \leftarrow fluctCog \wedge visHall \wedge not\ fn$
4. **probable**: $DLB \leftarrow fluctCog \wedge extraPyr \wedge not\ fn$
5. **probable**: $VaD \wedge DLB \leftarrow fn \wedge radVasc \wedge extraPyr \wedge fluctCog$

6. **possible**: $VaD \wedge DLB \leftarrow fn \wedge fluctCog$
7. **possible**: $VaD \wedge AD \leftarrow fn \wedge slow \wedge prog \wedge epiMem$
8. **possible**: $VaD \wedge AD \leftarrow radVasc \wedge slow \wedge prog \wedge epiMem$
9. **possible**: $DLB \wedge AD \leftarrow fluctCog \wedge slow \wedge prog \wedge epiMem$
10. **possible**: $DLB \wedge AD \leftarrow extraPyr \wedge slow \wedge prog \wedge epiMem$
11. **possible**: $DLB \wedge AD \leftarrow visHall \wedge slow \wedge prog \wedge epiMem$
12. **possible**: $DLB \leftarrow fluctCog$
13. **possible**: $DLB \leftarrow visHall$
14. **possible**: $DLB \leftarrow extraPyr$
15. **possible**: $VaD \leftarrow fn$
16. **possible**: $VaD \leftarrow radVasc$
17. **supported**: $VaD \leftarrow fluctCog$
18. **plausible**: $VaD \leftarrow fn$
19. **probable**: $AD \leftarrow slow \wedge prog \wedge epiMem \wedge \ not\ (VaD \vee DLB)$

A problem in the dementia domain is that a large number of symptoms are overlapping between diseases. In addition, it is common to have more than one disease causing dementia in old age and in later stages of the disease progression (comorbidity). Typically, formal representations do not support this kind of complexity of a differential diagnostic process. The advantage of applying possibilistic nested rules is that it provides a transparent method to capture the different ways to interpret a set of findings, including potential comorbidity. Transparency is highly desirable in a knowledge modeling situation where medical domain experts are responsible for the content. Our example exemplify this, showing that one of two possible medical conditions may be present, or both.

3.2 Possibilistic Nested Logic Semantics

In order to define the semantics of the possibilistic nested logic programs, we introduce some basic concepts with respect to sets of possibilistic atoms.

Given a finite set of atoms $\mathcal{A}$, a lattice $(\mathcal{Q},\leq)$ and a the function *Cardinality* which returns the cardinality of a set:

$$PS = \{S | S \in 2^{\mathcal{A}\times\mathcal{Q}} \text{ and } \forall x \in \mathcal{A}, Cardinality(\{(x,\alpha)|(x,\alpha) \in S\}) \leq 1\}$$

Observe that every $S \in PS$ is a set of possibilistic atoms where every atom $x \in \mathcal{A}$ at most occurs one time in S.

▶ **Definition 8.** [14] Let $\mathcal{A}$ be a finite set of atoms and $(\mathcal{Q},\leq)$ be a lattice. $\forall A, B \in \mathcal{PS}$, we define

$$A \sqcap B = \{(x, \mathcal{GLB}(\{\alpha,\beta\}))|(x,\alpha) \in A \wedge (x,\beta) \in B\}.$$

$$\begin{aligned} A \sqcup B = {} & \{(x,\alpha)|(x,\alpha) \in A \textit{ and } x \notin B^*\} \cup \\ & \{(x,\alpha)|x \notin A^* \textit{ and } (x,\alpha) \in B\} \cup \\ & \{(x, \mathcal{LUB}(\{\alpha,\beta\}))|(x,\alpha) \in A \text{ and } \ (x,\beta) \in B\}. \end{aligned}$$

$$\begin{aligned} A \sqsubseteq B \iff {} & A^* \subseteq B^*, \text{ and } \forall x,\alpha,\beta,(x,\alpha) \in A \wedge \\ & (x,\beta) \in B \text{ then } \alpha \leq \beta. \end{aligned}$$

Before moving on, let us define the concept of *i-greatest set w.r.t.* $\mathcal{PS}$ as follows: Given $M \in \mathcal{PS}$, M is an *i-greatest set* in $\mathcal{PS}$ iff $\nexists M' \in \mathcal{PS}$ such that $M \sqsubseteq M'$. For instance, let $\mathcal{PS} = \{\{\{(a,2),(b,1)\},\{(a,2),(b,2)\}\}$. One can see that $\mathcal{PS}$ has one i-greatest sets: $\{(a,2),(b,2)\}$.

Similar to the definition of answer set semantics for nested logic programs, the possibilistic answer set semantics for possibilistic nested logic programs is defined in terms of a syntactic reduction.

▶ **Definition 9** (Reduction P_M)**.** Let $P = \langle(\mathcal{Q}, \leq), N\rangle$ be a possibilistic nested logic program, M be a set of atoms. P reduced by M is the following possibilistic definite nested logic program:

$$P_M := \{\alpha : (A \leftarrow B)^M | \alpha : A \leftarrow B \in N \text{ and } M \text{ is closed under } (A \leftarrow B)^M\}$$

Observe that the reduction $(A \leftarrow B)^M$ is according to Definition 4 and P_M is a possibilistic definite nested logic programs.

Now by considering the inference of possibilistic logic ($\vdash_{PL}$) and the reduction P_M, the inference relation $\Vdash_{PL}$ is defined as follows:

▶ **Definition 10.** Let $P = \langle(\mathcal{Q}, \leq), N\rangle$ be a possibilistic nested logic program and $M \in \mathcal{PS}$.
We write $P \Vdash_{PL} M$ when M^* is an answer set of P^* and $P_{M^*} \vdash_{PL} M$.

Observe that the inference relation $\Vdash_{PL}$ is considering the standard definition of answer sets for nested logic programs (Definition 6). In particular, $\Vdash_{PL}$ is identifying sets of possibilistic atoms which satisfy P. However, not all these sets are optimal in the sense of necessity-values of a possibilistic theory. Hence, in order to define the possibilistic answer sets of a possibilistic nested logic programs we consider the idea of an *i-greatest set*.

▶ **Definition 11.** Let $P = \langle(\mathcal{Q}, \leq), N\rangle$ be a possibilistic nested logic program and M be a set of possibilistic atoms. M is a possibilistic answer set of P iff M is an i-greatest set in $\mathcal{PS}$ such that $P \Vdash_{PL} M$. $NSEM(P)$ denotes the set of possibilistic answer sets of P.

In order to illustrate the definition of answer sets for possibilistic nested logic programs, let us consider a subset of possibilistic nested rules which were introduced in Example 7.

▶ **Example 12.** Let $P = \langle(\mathcal{Q}, \leq), N\rangle$ be a possibilistic nested logic program in which $(\mathcal{Q}, \leq)$ is the lattice introduced in Example 7 and N is the following set of possibilistic nested rules:

$$\begin{array}{ll}
\textit{confirmed}: & \textit{fn} \leftarrow \top \\
\textit{confirmed}: & \textit{radVasc} \leftarrow \top \\
\textit{confirmed}: & \textit{extraPyr} \leftarrow \top \\
\textit{confirmed}: & \textit{fluctCog} \leftarrow \top \\
\textit{probable}: & \textit{VaD} \wedge \textit{DLB} \leftarrow \textit{fn} \wedge \textit{radVasc} \wedge \\
 & \textit{extraPyr} \wedge \textit{fluctCog} \\
\textit{possible}: & \textit{DLB} \leftarrow \textit{extraPyr} \\
\textit{probable}: & \textit{VaD} \leftarrow \textit{fn} \wedge \textit{radVasc} \wedge \\
 & \textit{not}\ (\textit{AD} \vee \textit{DLB})
\end{array}$$

In order to infer the answer sets of P, the first step is to find, the answer set of P^*. It is not hard to see that P^* has only one answer set which is $M = \{$ *fn,radVasc,extraPyr,fluctCog,DLB,VaD* $\}$. Now, one can see that P_M is:

$$\begin{array}{ll}
\textit{confirmed}: & \textit{fn} \leftarrow \top \\
\textit{confirmed}: & \textit{radVasc} \leftarrow \top \\
\textit{confirmed}: & \textit{extraPyr} \leftarrow \top \\
\textit{confirmed}: & \textit{fluctCog} \leftarrow \top \\
\textit{probable}: & \textit{VaD} \wedge \textit{DLB} \leftarrow \textit{fn} \wedge \textit{radVasc} \wedge \\
 & \textit{extraPyr} \wedge \textit{fluctCog} \\
\textit{possible}: & \textit{DLB} \leftarrow \textit{extraPyr}
\end{array}$$

Observe that the possibilistic nested rule $r = probable : VaD \leftarrow fn \wedge radVasc \wedge not\ (AD \vee DLB)$ was removed because $(r^*)^M$ is not closed under M. Now let us consider $M_1 = \{(fn, confirmed), (radVasc, confirmed), (extraPyr, confirmed), (fluctCog, confirmed), (DLB, probable), (VaD, probable)\}$ and $M_2 = \{(fn, confirmed), (radVasc, confirmed), (extraPyr, confirmed), (fluctCog, confirmed), (DLB, possible), (VaD, probable)\}$.

One can see that $P_M \vdash_{PL} M_1$ and $P_M \vdash_{PL} M_2$. Since $M = M_1^* = M_2^*$, hence both M_1^* and M_2^* are answer sets of P^*. Therefore $P_M \Vdash_{PL} M_1$ and $P_M \Vdash_{PL} M_2$. This means that both M_1 and M_2 are two potential sets to be answer sets of P. Observe that $M_2 \sqsubseteq M_1$, therefore M_2 is not an i-greatest set. One can see that M_1 is an i-greatest set, therefore M_1 is the unique possibilistic answer set of P.

An obvious property of the logic programming semantics of the possibilistic nested logic programs is that it generalizes the logic programming semantics of nested logic programs

▶ Proposition 1. Let $P = \langle(\mathcal{Q}, \leq), N\rangle$ be a possibilistic nested logic program. If M is a possibilistic answer set of P then M^* is an answer set of P^*.

In the family of possibilistic logic programs, the approach presented in this paper generalizes the approaches presented in [13] and [14].

Let us formalize the relationship between the nested possibilistic semantics and the possibilistic stable semantics. The last one was introduced by [13].

▶ Proposition 2. Let $P = \langle(\mathcal{Q}, \leq), N\rangle$ be a possibilistic nested logic program such that for all $r \in N$, $r = \alpha : A_0 \leftarrow A_1 \wedge \cdots \wedge A_j \wedge\ not\ A_{j+1} \wedge \cdots \wedge not\ A_n$, $\mathcal{L}_{N^*}$ has no extended atoms and $(\mathcal{Q}, \leq)$ is a total ordered set. If M is a consistent possibilistic answer set of P then M is a possibilistic stable model according to the definition from [13].

Now, let us show that the possibilistic semantics for possibilistic nested logic programs generalizes the semantics of possibilistic disjunctive logic programs.

▶ Proposition 3. Let $P = \langle(\mathcal{Q}, \leq), N\rangle$ be a possibilistic nested logic program such that for all $r \in N$, $r = \alpha : A_0 \vee \cdots \vee A_m \leftarrow A_{m+1} \wedge \ldots A_j \wedge not\ A_{j+1} \wedge \cdots \wedge\ not\ A_n$ in which $A_i (0 \leq i \leq n)$ are literals. If M is a consistent possibilistic answer set of P then M is a possibilistic answer set according to the definition from [14].

It is known that the answer set semantics for nested logic programs is computable [10]. Indeed, one can find solvers of nested logic programs [19]. On the other hand, the possibilistic inference of possibilistic logic is complete and sound by a possibilistic extended version of the classical resolution rule [6]. Hence, it is not difficult to define an algorithm for computing the possibilistic answer sets of a possibilistic nested logic program.

A common strategy for computing the answer set of a nested logic program is to translate the nested logic programs into disjunctive ones. Hence, the answer sets of the nested logic programs are characterized by the answer sets of disjunctive logic programming systems. This strategy can be also applied for computing the answer sets of possibilistic nested logic programs via possibilistic disjunctive logic programs.

By lack of space, we omit the details of the transformation of any possibilistic nested logic program into a possibilistic logic program. The details of this transformation will be presented in the long version of this paper. In the following theorem, it is assumed that there is a transformation of any possibilistic nested logic program into a possibilistic disjunctive logic program.

▶ **Theorem 13.** *Let $P = \langle(\mathcal{Q}, \leq), N\rangle$ be a possibilistic nested logic program and P' a possibilistic disjunctive logic program obtained by transforming P. If M' is an answer set of P' then $M = \{(a, \alpha) | (a, \alpha) \in M'$ and $a \in M'^* \cap \mathcal{L}_{P^*}\}$ is answer set of P.*

4 Conclusions and Future Work

In the logic programming literature, one can find different approaches for expressing uncertain information [8, 13, 4, 1, 21, 14, 5]; however, most of them define syntactic restriction to their specification languages. Against this background, we introduce the class of possibilistic nested logic programs. The syntax and semantics of these programs generalize previous works in the paradigm of Answer Set Programming plus Possibilistic Logic (Proposition 2, Proposition 3). Moreover, our approach generalizes the frame of nested logic programs (Proposition 1). We show that the semantics of the possibilistic nested programs can be computed by transforming possibilistic nested logic programs into possibilistic disjunctive logic programs (Theorem 13).

In the long version of this paper, we will present a process for transforming a possibilistic nested logic program into a possibilistic disjunctive logic program. In this process, we will identify the class of *possibilistic generalized disjunctive logic programs* which is a subclass (syntactically speaking) of the possibilistic nested logic programs. Let us observe that the class of possibilistic generalized disjunctive logic programs is a class of possibilistic programs which is interesting by itself due to this class of logic programs is the possibilistic extension of the generalized disjunctive logic programs explored in [9].

To the best of our knowledge, the approach presented in this paper is the first work to attend to manage uncertain information with no-syntactic restrictions in its rules. It is worth mentioning that the possibilistic nested logic programs combine both non-monotonic reasoning and reasoning under uncertainty in a single framework.

Since the uncertain information in possibilistic nested logic programs can be captured by partially ordered sets, the possibilistic nested programs define a suitable approach for capturing qualitative information. In particular, we have illustrated that possibilistic nested logic programs are expressive enough for capturing ambiguous and uncertain knowledge content in medical guidelines. The approach has the potential to provide medical experts, who are usually not experts in knowledge representation, with a formal framework that is transparent and intuitive for knowledge modeling.

In our future, we will explore practical algorithms for implementing a solver for possibilistic nested logic programs. It is worth mentioning that there already exist solvers of nested logic programs [19]; hence, a solver for nested logic programs can be taken as a starting point for a solver for possibilistic nested logic programs. The approach described in this paper will be evaluated in practical knowledge modeling and diagnostic situations involving medical professionals as part of the ACKTUS project [11].

5 Acknowledgements

This research has been supported by VINNOVA (The Swedish Governmental Agency for Innovation Systems) and the Swedish Brain Power.

References

1 Teresa Alsinet, Carlos Iván Chesñevar, Lluis Godo, and Guillermo Ricardo Simari. A logic programming framework for possibilistic argumentation: Formalization and logical properties. *Fuzzy Sets and Systems*, 159(10):1208–1228, 2008.

2 American Psychiatric Association. *Diagnostic and Statistical Manual of Mental Disorders DSM-IV-TR Fourth Edition (Text Revision)*. Amer Psychiatric Pub, 4th edition, 2000.

3 Chitta Baral. *Knowledge Representation, Reasoning and Declarative Problem Solving*. Cambridge University Press, Cambridge, 2003.
4 Chitta Baral, Michael Gelfond, and J. Nelson Rushton. Probabilistic reasoning with answer sets. *TPLP*, 9(1):57–144, 2009.
5 Kim Bauters, Steven Schockaert, Martine De Cock, and Dirk Vermeir. Weak and strong disjunction in possibilistic asp. In *SUM*, volume 6929 of *Lecture Notes in Computer Science*, pages 475–488. Springer, 2011.
6 Didier Dubois, Jérôme Lang, and Henri Prade. Possibilistic logic. In Dov Gabbay, Christopher J. Hogger, and J. A. Robinson, editors, *Handbook of Logic in Artificial Intelligence and Logic Programming, Volume 3: Nonmonotonic Reasoning and Uncertain Reasoning*, pages 439–513. Oxford University Press, Oxford, 1994.
7 Michael Gelfond and Vladimir Lifschitz. Classical Negation in Logic Programs and Disjunctive Databases. *New Generation Computing*, 9:365–385, 1991.
8 Michael Kifer and V. S. Subrahmanian. Theory of generalized annotated logic programming and its applications. *J. Log. Program.*, 12(3&4):335–367, 1992.
9 V. Lifschitz. *Principles of Knowledge Representation*, chapter Foundations of Logic Programming, pages 69–128. CSLI Publications, 1996.
10 Vladimir Lifschitz, Lappoon R. Tang, and Hudson Turner. Nested expressions in logic programs. *Annals of Mathematics and Artificial Intelligence*, 25(3-4):369–389, 1999.
11 Helena Lindgren and Peter Winnberg. Evaluation of a semantic web application for collaborative knowledge building in the dementia domain. In *eHealth*, pages 62–69, 2010.
12 Peter Lucas. Symbolic diagnosis and its formalisation. *The Knowledge Engineering Review*, 12:109–146, 1997.
13 Pascal Nicolas, Laurent Garcia, Igor Stéphan, and Claire Lefèvre. Possibilistic Uncertainty Handling for Answer Set Programming. *Annals of Mathematics and Artificial Intelligence*, 47(1-2):139–181, 2006.
14 Juan Carlos Nieves, Mauricio Osorio, and Ulises Cortés. Semantics for Possibilsitic Disjuntive Programs. *Theory and Practice of Logic Programming*, Available on doi: 10.1017/S1471068411000408, 2011.
15 J O'Brien, D Ames, and A Burns, editors. *Dementia*. Arnold, 2000.
16 Mauricio Osorio, Juan Antonio Navarro Pérez, and José Arrazola. Applications of intuitionistic logic in answer set programming. *TPLP*, 4(3):325–354, 2004.
17 David Pearce. Stable inference as intuitionistic validity. *J. Log. Program.*, 38(1):79–91, 1999.
18 Henri Prade. Advances in data management. In *Current Research Trends in Possibilistic Logic: Multiple Agent Reasoning, Preference Representation, and Uncertain Databases*. Springer Berlin / Heidelberg, 2009.
19 Vladimir Sarsakov, Torsten Schaub, Hans Tompits, and Stefan Woltran. nlp: A compiler for nested logic programming. In *LPNMR*, Lecture Notes in Computer Science, pages 361–364. Springer, 2004.
20 David Silverman. *Interpreting Qualitative Data*. SAGE Publications, 2006.
21 Davy Van-Nieuwenborgh, Martine De Cock, and Dirk Vermeir. An introduction to fuzzy answer set programming. *Ann. Math. Artif. Intell.*, 50(3-4):363–388, 2007.