Modeling Epistemic Knowledge about Users

MIRCEA PREDA
University of Craiova
Department of Computer Science
Al. I. Cuza street 13, 200585 Craiova
Romania

Abstract: Representing what an intelligent agent knows or believes can be an important feature for an entity that acts in a collaborative environment. Epistemic logic programs are a clear way to reason about what an intelligent agent knows or believes, but the complexity of the reasoning is exponential. The paper presents a new kind of logic programs, the logic programs with two types of negation as failure operators. An answer set semantics is defined for these programs and it is proved that this semantics allows representing epistemic knowledge. Moreover, a polynomial approximation for this semantics is constructed and the approximation maintains the epistemic knowledge representation possibilities. The paper concludes with examples of knowledge representation resolved using the new proposed type of logic programs. They show how the logic programs can be integrated in a human computer interface.

Key–Words: Knowledge representation, epistemic logic programs, answer set semantics, well founded semantics, human computer interface.

1 Introduction

Epistemic logic programs ([3], [9]) are a particular type of logic programs that are able to represent epistemic information about an intelligent agent. Epistemic information is particular important for applications that interact with human users. For example, for a Web recommender application is important to figure what knows or believes a human visitor in order to provide accurate recommendations[7]. Knowledge about visitors can be represented by logic programs and, by performing logic reasoning, a Web recommender can establish what knows or believes a particular visitor. Similarly, in an e-learning application is important to realize what knows or believes a student in order to adjust the learning process to his needs. Unfortunately, epistemic logic programs have a serious drawback, the complexity of the reasoning is exponential regarding the number of atoms from a epistemic logic program.

In this paper, we propose a new type of logic programs, the logic programs with two type of negation as failure operators and show that these programs are able to clearly and efficiently represent epistemic information. First, the syntax and the answer set semantics of the logic programs with two negations (*LP2N*) are defined and it is proved that the answer set semantics allows representing what an intelligent agent knows or believes. After that, a polynomial time approximation for the answer set semantics is constructed and it is shown that the approximation maintains the epistemic capabilities of the answer set semantics. Several examples of knowledge representation using these programs are provided. The paper concludes with a summary of the obtained results.

These developments are an extension of the work presented in [5] and [6]. The logic programs introduced by these papers can use an arbitrary number of negation as failure operators but they are more difficult to understand and interpret. Also, the polynomial approximation results are weaker than the newly exposed ones.

2 Mathematical background

Definition 1 *Let us consider a partial ordered set* $(L, <)$. $(L, <)$ *is a lattice if and only if* $\forall x, y \in L$ *the set* $\{x, y\} \subseteq L$ *has an infimum and a supremum.* $(L, <)$ *is a complete lattice if every subset* $A \subseteq L$ *has an infimum and a supremum.*

Definition 2 *Let* $(L, <)$ *a partial ordered set and* $f : L \to L$ *a function.* f *is called monotone if and only if* $\forall x, y \in L,\ x < y$ *then* $f(x) < f(y)$. f *is called anti monotone if and only if* $\forall x, y \in L,\ x < y$ *then* $f(y) < f(x)$.

Theorem 3 *(Knaster-Tarski theorem [4]) Let* $(L, <)$ *be a complete lattice and let* $f : L \to L$ *a monotone*

function. Then the set of the fix points of $f(.)$ in L is also a complete lattice.

Remark 4 *Since complete lattices cannot be empty, the theorem in particular guarantees the existence of at least one fixed point of $f(.)$, and even the existence of a least (or greatest) fixed point. The smallest fix point of $f(.)$ is the smallest element x with the property that $f(x) = x$ or, equivalently, $f(x) < x$. The greatest fix point of $f(.)$ is the greatest element x with the property that $f(x) = x$.*

Definition 5 *A definite logic program Π is a collection of rules*

$$C \leftarrow A_1, ..., A_m$$

where $m \geq 0$ and $C, A_1, ..., A_m$ are ground atoms. The set of the all atoms from a definite logic program Π is called Herbrand Base and denoted by $HB(\Pi)$. The minimal model of a definite logic program Π, denoted by $m(\Pi)$, is the smallest subset of $HB(\Pi)$ with the property that for every rule $C \leftarrow A_1, ..., A_m$ if $C \in m(\Pi)$ then $\{A_1, ..., A_m\} \subseteq m(\Pi)$.

3 Logic programs with two negation operators

Let us consider two operators of negation as failure: not_1 and not_2.

Definition 6 *Formally, by a logic program with two negation operators (LP2N), we mean a collection(set) of rules that can have two forms:*

$$\begin{array}{rcl} C & \longleftarrow & A_1, ..., A_m, not_1 B_1, ..., not_1 B_n \\ C & \longleftarrow & A_1, ..., A_m, not_2 B_1, ..., not_2 B_n \end{array}$$

where $m, n \geq 0$, C, As, Bs are ground atoms. The expression to the left of $\longleftarrow$ is called the head of the rule, while the expression to the right of $\longleftarrow$ is called the body of the rule.

Definition 7 *Let Π be a LP2N. By $ord(\Pi)$, we denote the greatest index of the negation as failure operators that appear in Π. $ord(\Pi) \leq 2$. If Π does not contain negation as failure then $ord(\Pi) = 0$.*

Definition 8 *(an extension of Gelfond-Lifschitz transformation [2]) Let Π be a LP2N and $ord(\Pi) = k \geq 1$. For any set S of literals, we denote by Π^S the LP2N obtained from Π by deleting*

(I) each rule that has a formula $not_k B$ in its body with $B \in S$

(II) all formulas of the form $not_k B$ in the bodies of the remaining rules.

$ord(\Pi^S) \leq k - 1$.

Definition 9 *Let Π be a LP2N such that $ord(\Pi) = 0$. The answer set of Π is the smallest (in the sense of the set-theoretic inclusion) subset $S \subseteq HB(\Pi)$ such that for any rule $C \longleftarrow A_1, ..., A_m$ from Π if $\{A_1, ..., A_m\} \subseteq S$ then $C \in S$. Let Π be a LP2N and $ord(\Pi) \geq 1$. We say that $S \subseteq HB(\Pi)$ is an answer set of Π if and only if*

$$S = \bigcap_{S' \in AS(\Pi^S)} S'$$

where, for any LP2N Π, by $AS(\Pi)$ we denote the family of the all answer sets of Π.

An answer set should be viewed as a possible state of the world. Let Π be an *LP2N* and $S \in AS(\Pi)$ an answer set. The answer of the pair (Π, S) to an atom query q is *Yes* under answer set semantics if $q \in S$ and *No* if $q \notin S$. We will say that, regarding to a *LP2N* Π, an atom a can be considered known if $a \in \bigcap_{S \in AS(\Pi)} S$ (a appears in the all possible states of the world) and a can be believed if $a \in \bigcup_{S \in AS(\Pi)} S$ (a appears in at least one possible state of the world).

Proposition 10 *Let Π be a LP2N with $ord(\Pi) = 1$ and $a \in HB(\Pi)$. We build the LP2N Π' adding to Π the rules*

$$\begin{array}{l} p \longleftarrow not_1\ a \\ q \longleftarrow not_2\ p \end{array}$$

where p,q are new atoms, $p, q \notin HB(\Pi)$.
In these conditions, the following equivalences are true:

(I) $\exists S \in AS(\Pi)$ such that $a \in S \Longleftrightarrow \Pi' \models q$ (i.e. Π''s answer to the query q is yes).

(II) $\forall S \in AS(\Pi)$, $a \notin S \Longleftrightarrow \Pi' \models p$.

Proof: We will prove that Q is an answer set of Π' if and only if

$$Q = \begin{cases} \bigcap\limits_{S \in AS(\Pi)} S \cup \{p\} \text{ if } a \notin \bigcup\limits_{S \in AS(\Pi)} S \\ \bigcap\limits_{S \in AS(\Pi)} S \cup \{q\} \text{ if } a \in \bigcup\limits_{S \in AS(\Pi)} S \end{cases}.$$

"$\Rightarrow$" Consider an answer set $Q \in AS(\Pi')$, $Q = \bigcap_{S' \in AS(\Pi'^Q)} S'$. There are two cases.

Case 1. $p \in Q$. Then $\Pi'^Q = \Pi \cup \{p \longleftarrow not_1\ a\}$. The following equivalence is obvious: $S \in AS(\Pi)$

iff $S' \in AS(\Pi'^Q)$ where $S' = \begin{cases} S \text{ if } a \in S \\ S \cup \{p\} \text{ if } a \notin S \end{cases}$.
Therefore,

$$\bigcap_{S' \in AS(\Pi'^Q)} S' = \begin{cases} \bigcap\limits_{S \in AS(\Pi)} S \text{ if } a \in \bigcup\limits_{S \in AS(\Pi)} S \\ \bigcap\limits_{S \in AS(\Pi)} S \cup \{p\} \text{ if } a \notin \bigcup\limits_{S \in AS(\Pi)} S \end{cases}$$

But $p \in Q$, therefore $a \notin \bigcup\limits_{S \in AS(\Pi)} S$.

Case 2. $p \notin Q$. Then $\Pi'^Q = \Pi \cup \{p \longleftarrow not_1\ a, q \longleftarrow\}$. The following equivalence is obvious: $S \in AS(\Pi)$ if and only if $S' \in AS(\Pi'^Q)$ where $S' = \begin{cases} S \cup \{q\} \text{ if } a \in S \\ S \cup \{p, q\} \text{ if } a \notin S \end{cases}$. Therefore

$$\bigcap_{S' \in AS(\Pi'^Q)} S' = \begin{cases} \bigcap\limits_{S \in AS(\Pi)} S \cup \{q\} \text{ if } a \in \bigcup\limits_{S \in AS(\Pi)} S \\ \bigcap\limits_{S \in AS(\Pi)} S \cup \{p, q\} \text{ if } a \notin \bigcup\limits_{S \in AS(\Pi)} S \end{cases}$$

But $p \notin Q$, therefore $a \in \bigcup\limits_{S \in AS(\Pi)} S$.

"$\Leftarrow$" Case 1. Consider $Q = \bigcap\limits_{S \in AS(\Pi)} S \cup \{p\}$ and $a \notin \bigcup\limits_{S \in AS(\Pi)} S$. Then $\Pi'^Q = \Pi \cup \{p \longleftarrow not_1\ a\}$. We obtain that $\bigcap\limits_{S' \in AS(\Pi'^Q)} S' = \bigcap\limits_{S \in AS(\Pi)} S \cup \{p\} = Q$, therefore $Q \in AS(\Pi')$.

Case 2. Consider $Q = \bigcap\limits_{S \in AS(\Pi)} S \cup \{q\}$ and $a \in \bigcup\limits_{S \in AS(\Pi)} S$. Then $\Pi'^Q = \Pi \cup \{p \longleftarrow not_1\ a, q \longleftarrow\}$. We obtain that $\bigcap\limits_{S' \in AS(\Pi'^Q)} S' = \bigcap\limits_{S \in AS(\Pi)} S \cup \{q\} = Q$, therefore $Q \in AS(\Pi')$.

Remark 11 *Π' has only one answer set.*

Remark 12 *It is natural to say that an intelligent agent, represented by a set of premises Π, may believe that an atom a is true if and only a occurs in at least one answer set of Π. The operator M (believe operator) defined for epistemic logic programs [3] can be incorporated in this way into* LP2N*s. The newly introduced atoms p and q indicate us if a occurs in at least one answer set of Π or not.*

Proposition 13 *Let Π be a LP2N with $ord(\Pi) = 1$ and $a \in HB(\Pi)$. We build the program Π' adding to Π the rule*

$$p \longleftarrow not_2\ a$$

where p is a new atom, $p \notin HB(\Pi)$. In these conditions, the following equivalences are true:

(I) $a \in S, \forall S \in AS(\Pi) \Longleftrightarrow \Pi' \models a$

(II) $\exists S \in AS(\Pi)$ *such that* $a \notin S \Longleftrightarrow \Pi' \models p$.

Proof: The proof is based on the equivalence: Q is an answer set of Π' if and only if

$$Q = \begin{cases} \bigcap\limits_{S \in AS(\Pi)} S \text{ if } a \in \bigcap\limits_{S \in AS(\Pi)} S \\ \bigcap\limits_{S \in AS(\Pi)} S \cup \{p\} \text{ if } a \notin \bigcap\limits_{S \in AS(\Pi)} S \end{cases}.$$

This proposition allows to incorporate the operator K (know operator) defined for epistemic logic programs into *LP2N*s. It is natural to say that an intelligent agent, with a set of premises Π, knows that an atom a is true if and only if a occurs in all answer sets of Π. The newly introduced atom p indicates if a does not occur in all answer sets of Π.

4 An approximation for the answer set semantics

The answer set semantics provides us intuitive answers but in practice its exponential complexity is a major drawback.

Let us consider the function $f_\Pi : HB(\Pi) \rightarrow HB(\Pi)$

$$f_\Pi(S) = \begin{cases} m(\Pi) \text{ if } ord(\Pi) = 0 \\ \inf\{S' | S' = f_{\Pi^S}(S')\} \text{ otherwise} \end{cases},$$

where Π is a *LP2N*, $m(\Pi)$ is the minimal model of the *LP2N* Π and $\inf\{S' | S' = f_{\Pi^S}(S')\} = \bigcap_{S' = f_{\Pi^S}(S')} S'$.

A *LP2N* Π can be divided in three parts: Π_0 is the set of rules of Π that do not contain the negation as failure, Π_1 is the set of rules of Π that contain the not_1 negation as failure operator and, similarly, Π_2 includes the rules that contain the not_2 operator.

Definition 14 *A LP2N Π is named stratified if and only if the following condition hold:*

$$head(\Pi_2) \cap body(\Pi_1 \cup \Pi_0^1) = \emptyset$$

where $head(\Pi)$ is the set of the atoms which occur in the heads of rules from Π, $body(\Pi)$ is the set of the atoms from the bodies of rules from Π and $\Pi_0^1 \subseteq \Pi_0$ is the smallest subset of rules of Π_0 which satisfies the following properties:
- it contains all rules of Π_0 with the property that their heads occur in a body of rule from Π_1
- if the head of a rule from Π_0 occurs in the body of a rule from Π_0^1 then this rule belongs to Π_0^1.

Lemma 15 *Let Π be a stratified LP2N with $ord(\Pi) = k \leq 2$. Then, the application $f_\Pi(.)$ is anti-monotone.*

Proof: Let $S_1 \subseteq S_2 \subseteq HB(\Pi)$. It can be proved that $f_\Pi(S_1) \supseteq f_\Pi(S_2)$.

If $ord(\Pi) = 0$ then $f_\Pi(S_1) = f_\Pi(S_2) = m(\Pi)$.

If $ord(\Pi) = 1$ then $\Pi^{S_1} \supseteq \Pi^{S_2}$, $ord(\Pi^{S_1}) = ord(\Pi^{S_2}) = 0$, $m(\Pi^{S_1}) \supseteq m(\Pi^{S_2})$ and, consequently, $f_\Pi(S_1) \supseteq f_\Pi(S_2)$.

If $ord(\Pi) = 2$ then $\Pi^{S_1} \supseteq \Pi^{S_2}$, $ord(\Pi^{S_1}) = ord(\Pi^{S_2}) \leq 1$. The following relations are true:

$$\begin{aligned} \Pi^{S_1} &= \Pi_2^{S_1} \cup (\Pi_1 \cup \Pi_0), \\ \Pi^{S_2} &= \Pi_2^{S_2} \cup (\Pi_1 \cup \Pi_0), \\ \Pi_2^{S_1} &\supseteq \Pi_2^{S_2}, \\ \Pi^{S_1} \setminus \Pi^{S_2} &= \Pi_2^{S_1} \setminus \Pi_2^{S_2}, \\ 0 &= ord(\Pi^{S_1} \setminus \Pi^{S_2}). \end{aligned}$$

Due to stratification,

$$head(\Pi^{S_1} \setminus \Pi^{S_2}) \cap (\Pi_1 \cup \Pi_0^1) = \emptyset. \quad (1)$$

Let S' be a fix point of the function $f_{\Pi^{S_1}}(.)$, $f_{\Pi^{S_1}}(S') = S'$. There are two cases.

Case 1. $ord(\Pi^{S_1}) = 0$. Then $ord(\Pi^{S_2}) = 0$. $f_{\Pi^{S_1}}(S') = m(\Pi^{S_1})$, so $S' = m(\Pi^{S_1})$. There is $S'' = m(\Pi^{S_2})$ such that $S'' \subseteq S'$ and $S'' = f_{\Pi^{S_2}}(S'')$.

Case 2. $ord(\Pi^{S_1}) = 1$. Then $ord(\Pi^{S_2}) = 1$.

$$\begin{aligned} f_{\Pi^{S_1}}(S') &= \bigcap_{S^* = f_{\Pi^{S_1 S'}}(S^*)} S^* \quad (2) \\ &= m(\Pi^{S_1 S'}) \quad (3) \\ &= m(\Pi_2^{S_1} \cup \Pi_1^{S'} \cup \Pi_0). \quad (4) \end{aligned}$$

Let S^* be such that $S^* = m(\Pi_2^{S_1} \cup \Pi_1^{S'} \cup \Pi_0)$ and S^{**} be the set obtained by removing from $m(\Pi_2^{S_1} \cup \Pi_1^{S'} \cup \Pi_0)$ the atoms generated by the rules from $\Pi_2^{S_1} \setminus \Pi_2^{S_2}$. Let S'' be S' without the atoms generated by $\Pi_2^{S_1} \setminus \Pi_2^{S_2}$. $S'' = \bigcap_{S^{**} = m(\Pi_2^{S_2} \cup \Pi_1^{S''} \cup \Pi_0)} S^{**}$, where S^{**} sets corresponds to the S^* sets from the formula 4. Results that $S'' \subseteq S'$ and $S'' = f_{\Pi^{S_2}}(S'')$.

We proved that for every fix point S' of $f_{\Pi^{S_1}}(.)$, $S' = f_{\Pi^{S_1}}(S')$ there is S'' a fix point of $f_{\Pi^{S_2}}(.)$, $S'' = f_{\Pi^{S_2}}(S'')$ such that $S'' \subseteq S'$. Consequently, $\inf\{S''|S'' = f_{\Pi^{S_2}}(S'')\} \subseteq \inf\{S'|S' = f_{\Pi^{S_1}}(S')\}$ and $f_\Pi(S_1) \supseteq f_\Pi(S_2)$.

Proposition 16 *Let Π be a stratified LP2N with $ord(\Pi) \leq 2$. The following relations are true:*

$$lfp(f_\Pi^2(.)) \subseteq \inf\{S|S = f_\Pi(S)\} = \bigcap_{S \in AS(\Pi)} S,$$

$$\bigcup_{S \in AS(\Pi)} S = \sup\{S|S = f_\Pi(S)\} \subseteq gfp(f_\Pi^2(.)).$$

Consequently, $lfp(f_\Pi^2(.))$ and $gfp(f_\Pi^2(.))$ are an approximation of the answer set semantics for the logic program Π.

Proof: $f_\Pi^2(.)$, is monotone and, according to Tarski theorem, it has smallest and greatest fix points. The fix points of $f_\Pi(.)$ are also fix points for $f_\Pi^2(.)$ and the theorem is proved.

5 Properties of the approximation for the answer sets semantics

The following two propositions show that using the approximation of the answer set semantics the *LP2N*s are still able to represent epistemic knowledge. The proof of these results can be obtaining extending the results from [6].

Proposition 17 *Let Π be a stratified LP2N with $ord(\Pi) \leq 2$ and a an atom, $a \in HB(\Pi)$. We build the LP2N Π' adding to Π the rule*

$$p \leftarrow not_2\ a$$

where p is a new atom, $p \notin HB(\Pi)$. Then

$$lfp(f_{\Pi'}^2(.)) = \begin{cases} lfp(f_\Pi^2(.)) \text{ if } a \in lfp(f_\Pi^2(.)) \\ lfp(f_\Pi^2(.)) \cup \{p\} \text{ if } a \notin gfp(f_\Pi^2(.)) \end{cases}$$

It is natural to define that an intelligent agent with a set of premises Π knows that an atom a is true if and only if $a \in lfp(f_{\Pi'}^2(.))$ regarding the approximation of the answer set semantics. An intelligent agent will not known that an atom a is true if $a \notin gfp(f_\Pi^2(.))$. The newly introduced atom p helps to determine when $a \notin gfp(f_\Pi^2(.))$.

Proposition 18 *Let Π be a stratified LP2N with $ord(\Pi) = 1$ and a an atom, $a \in HB(\Pi)$. We build the LP2N Π' adding to Π the rules:*

$$\begin{aligned} p &\leftarrow not_1\ a \\ q &\leftarrow not_2\ p \end{aligned}$$

where p, q are new atoms, $p, q \notin HB(\Pi)$. In these conditions

$$lfp(f_{\Pi'}^2(.)) = \begin{cases} \bigcap_{S \in AS(\Pi)} S \cup \{q\} \text{ if } a \in \bigcup_{S \in AS(\Pi)} S \\ \bigcap_{S \in AS(\Pi)} S \cup \{p\} \text{ if } a \notin \bigcup_{S \in AS(\Pi)} S \end{cases}$$

It is natural to define that an intelligent agent with a set of premises Π may believe that an atom a is true if and only if $q \in lfp(f_{\Pi'}^2(.))$ in relation with the approximation of the answer set semantics. If $q \in lfp(f_{\Pi'}^2(.))$ then it is guaranteed that a occurs in at least one answer set of Π.

6 Examples

Representing what agents know or believe. Consider the following information: "At a university, to promote the artificial intelligence(ai) course it is needed to read at least one of the books $book1$ and $book2$. If a student takes a course and he knows that a certain book is needed to be read then he buy the book. But, if a student takes a course and he may believe that a certain book is needed to be read at this course then he will search the book at library. John takes the ai course.". The epistemic program [3] that corresponds to our information, denoted by Π_{ep}, is:

$$take(john, ai) \leftarrow$$
$$need(ai, book1) \ or \ need(ai, book2) \leftarrow$$
$$buy(X, Y) \leftarrow take(X, Z), Kneed(Z, Y)$$
$$search(X, Y) \leftarrow take(X, Z), Mneed(Z, Y)$$

Here, the rules that contain variables are considered as representing the sets of their ground instantiations. By applying the previous results, the epistemic logic program can be transformed into the LP2N Π:

$$\begin{aligned}
need(ai, book1) &\leftarrow not_1\ need(ai, book2)\\
need(ai, book2) &\leftarrow not_1\ need(ai, book1)\\
take(john, ai) &\leftarrow\\
buy(X, Y) &\leftarrow take(X, Z),\\
&\quad know_need(Z, Y)\\
search(X, Y) &\leftarrow take(X, Z),\\
&\quad believe_need(Z, Y)\\
absent_need(Z, Y) &\leftarrow not_1\ need(Z, Y)\\
believe_need(Z, Y) &\leftarrow not_2\ absent_need(Z, Y)\\
nknow_need(Z, Y) &\leftarrow not_2\ need(Z, Y)\\
know_need(Z, Y) &\leftarrow not_2\ nknow_need(Z, Y)
\end{aligned}$$

Let us point that the disjunction operator *or* was replaced by two rules using negation as failure as is discussed in [8]. In these conditions we have $AS(\Pi) = \{\{search(john, book1), search(john, book2), believe\_need(ai, book1), believe\_need(ai, book2), nknow\_need(ai, book1), nknow\_need(ai, book2), take(john, ai)\}\}$ corresponding to our specifications. Π_{ep} is more compact because to epistemic logic programs the concepts "a reasoner may believe that ..." and "a reasoner knows that ..." are included in the semantics of the operators M and K. To LP2Ns these concepts are explicit defined in programs.

Representing the Unknown. The statements of the form "unknown p" cannot be proper represented by extended logic programs in the presence of multiple belief sets (answer sets). A detailed discussion on this topic is in [3] where the epistemic specifications are proposed as solution. More precise, a statement "unknown p" is represented there as a conjunction $\neg Kp$ and $\neg K\neg p$ where Kp stands for "p is known to be true by a reasoner based on the current set of premises ". An alternative view point can be the next. Let us consider a LP2N Π with $ord(\Pi) = 1$ as set of premises and an atom $p \in HB(\Pi)$. A statement "unknown p" is represented by the conjunction $not_2 p, not_2 \neg p$. The intended meaning is "we cannot prove p and $\neg p$ with the information from Π (with the information of the levels less or equal with 1)". Let us present this using an example from [3, 1]. Consider the LP2N Π:

$$eligible(X) \leftarrow highGPA(X)$$
$$eligible(X) \leftarrow minority(X), fairGPA(X)$$
$$\neg eligible(X) \leftarrow \neg fairGPA(X), \neg highGPA(X)$$
$$fairGPA(mike) \leftarrow not_1\ highGPA(mike)$$
$$highGPA(mike) \leftarrow not_1\ fairGPA(mike)$$

These rules represent the criteria used by a certain college for awarding scholarship and the situation of the student named "mike". We wish to represent the following information: "The students whose eligibility is not determined by the first three rules should be interviewed by the scholarship committee" i.e. $interview(X)$ if neither $eligible(X)$ nor $\neg eligible(X)$ is known. Consider the LP2N Π' obtained adding to Π the rule:

$$interview(X) \leftarrow not_2\ eligible(X), not_2\ \neg eligible(X)$$

We obtain $AS(\Pi') = \{\{interview(mike)\}\}$, $lfp(f^2_{\Pi'}(.)) = \{interview(mike)\}$, therefore Π' answers *yes* to the query $interview(mike)$ in the answer set semantics and regarding the approximation of the answer set semantics. Consequently, Π' has the intended behavior. The strong negation was handled by using the concept of positive form.

Representing user behavior related knowledge for a Web recommender system. Let us consider a Web recommender system for a digital library Web site that has hundreds of books organized in domains. *LP2N* programs are used in Web pages to describe what elements should be highlighted to the potential readers according with their state of knowledge. The following samples of epistemic reasoning can included:

- If a visitor has not complete information (does not known) about the presence of a general interest domain in the library then the recommender system must inform the visitor about the domain. The epistemic rule formalizing this piece of knowledge is

$$inform(X) \leftarrow \neg Kincluded\_domain(X), general\_interest\_domain(X).$$

- If a visitor has not complete information regarding the presence of a specialized domain in the

library and the recommender system establishes that the domain may be of interest for the visitor then it presents the domain. The associated epistemic rule is

$$inform(X) \leftarrow \neg Kincluded\_domain(X), specialized\_domain(X), Minterested(X).$$

- A visitor may be interested by a domain X if he/she visited a page with information about a related domain Y.

$$interested(X) \text{ or } \neg interested(X) \leftarrow visited(Y), related(X,Y).$$

- A visitor knows that a domain X is included in the library if he/she visited a page of a book Y from the domain.

$$included\_domain(X) \leftarrow visited(Y), domain(Y,X).$$

The programs attached to the Web pages describing books include their domains. For example, the program attached to the book "the_c_language_kernighan" includes the following two rules:

$$visited(the\_c\_language\_kernighan) \leftarrow$$
$$domain(the\_c\_language\_kernighan, programming\_languages) \leftarrow .$$

In some cases, a book cannot a have a clear specified domain. Different visitors can consider it included in a domain or not. For example, the book "win32_database_develop_guide" has attached the rule:

$$domain(win32\_database\_develop\_guide, multimedia\_databases) \text{ or}$$
$$\neg domain(win32\_database\_develop\_guide, multimedia\_databases) \leftarrow .$$

Obviously, a book can belong to more domains.

The above programs use the convenient syntax of the epistemic logic programs based on the operators K (know) and M (believe) [3]. These programs must be converted to $LP2N$s in order to benefit from the polynomial complexity result stated by proposition 16. The conversion will be based on propositions 17 and 18 and will be automatically performed by a logic programming support system. This logic support system has as purpose to compute the approximations of the answer set semantics for the logic programs included in the Web recommender system.

7 Conclusion

A new kind of logic programs with two types of negation as failure operators is proposed to represent knowledge about users. The answer set semantics for these programs and its polynomial approximation are able to represent epistemic information. Mathematical results indicate how to incorporate epistemic into *LP2N* and how transform epistemic logic programs into equivalent *LP2N*s. Several examples show how *LP2N*s can be used to incorporate knowledge in human computer interfaces. It is important to mention that the friendly syntax of the epistemic logic program [3] can still be used. Epistemic logic programs can be automatically translated in their *LP2N* counterparts.

Acknowledgments: The research was supported by the Romanian National University Research Council (AT grant No. 102/2007).

References:

[1] Chitta Baral and Michael Gelfond. Logic Programming and Knowledge Representation. Journal of Logic Programming, 19,20:73-148 (1994).

[2] Michael Gelfond and Vladimir Lifschitz. Classical Negation in Logic Programs and Disjunctive Databases. New Generation Computing, 9:365-385 (1991).

[3] Michael Gelfond. Logic Programming and Reasoning with Incomplete Information. Annals of Mathematics and Artificial Intelligence, 12:89-116 (1994).

[4] J. W. Lloyd. Foundations of Logic Programming. Springer-Verlag, second edition, 1987.

[5] M. Preda. Reasoning with epistemic information, Annals of the University of Craiova, Mathematics and Computer Science series, 1997, 24:166-186, ISSN: 1223-6934 (1997)

[6] M. Preda. A well founded semantics for logic programs with stratified negation, Annals of the University of Craiova, Mathematics and Computer Science series, 28:183-193, ISSN: 1223-6934 (2001).

[7] Mircea Preda and Dan Popescu. Personalized Web Recommendations: Supporting Epistemic Information about End-Users, Proc. of the 2005 IEEE/WIC/ACM Int. Conf. on Web Intelligence, Compiegne, France, IEEE Computer Press, ISBN: 0-7695-2415-X, pp. 692-695, (2005)

[8] Marco Schaerf. Negation and minimality in disjunctive databases. Journal of Logic Programming, (23):63-87, (1995).

[9] Yan Zhang, Epistemic Reasoning in Logic Programs, IJCAI-07, pp. 647-652, (2007)