ALLPAD: Approximate Learning of Logic Programs with Annotated Disjunctions

Fabrizio Riguzzi

Dipartimento di Ingegneria, Università di Ferrara, Via Saragat 1
44100 Ferrara, Italy,
friguzzi@ing.unife.it

Abstract. Logic Programs with Annotated Disjunctions (LPADs) provide a simple and elegant framework for representing probabilistic knowledge in logic programming. In this paper I consider the problem of learning ground LPADs starting from a set of interpretations annotated with their probability. I present the system ALLPAD for solving this problem. ALLPAD modifies the previous system LLPAD in order to tackle real world learning problems more effectively. This is achieved by looking for an approximate solution rather than a perfect one. ALLPAD has been tested on the problem of classifying proteins according to their tertiary structures and the results compare favorably with most other approaches.

1 Introduction

Logic Programs with Annotated Disjunctions (LPADs) [38, 37] are a relatively new formalism for representing probabilistic information in logic programming. They have been recognized as one of the simplest and most expressive [4] languages that combine logic and probability.

In [29] the definition of a learning problem for LPADs has been proposed together with an algorithm for solving it called LLPAD. However, LLPAD does not work well on non-toy problems because it relies on the exact solution of a large constraint satisfaction problem. On real world problems such a solution may not exist or may be too expensive to find. Therefore in this paper I propose the system ALLPAD (Approximate Learning of Logic Programs with Annotated Disjunctions) that modifies LLPAD in order to be able to solve real world problems by looking for a solution that "approximately" satisfies the learning problem. ALLPAD solves the constraint satisfaction problem in an approximate way by transforming it into an optimization problem. Moreover, the clause search phase is also modified in order to return only the most significant clauses so that the complexity of the optimization problem is kept inside acceptable limits.

To show that ALLPAD is able to work in practice, I have applied it to the problem of classifying proteins into SCOP classes. The accuracy obtained is compared to that of a naive Bayes approach and with the results of previous works. The comparison shows that the accuracy is significantly better than naive Bayes and that it is superior to all previous approaches apart from [15].

The paper is organized as follows. Section 2 provides some preliminary notions regarding LPADs together with their semantics as given in [38]. In section 3 I discuss two properties of ground LPADs that are exploited by LLPAD. In section 4 I introduce the learning problem and I describe LLPAD and ALLPAD. Section 5 shows under what conditions ALLPAD is correct. Section 7 presents the experiments performed in the SCOP domain. In section 8 I discuss related works and in section 9 I conclude and present directions for future work.

2 Preliminaries

2.1 LPADs

A disjunctive logic program [23] is a set of disjunctive clauses. A disjunctive clause is a formula of the form

$$h_1 \vee h_2 \vee \ldots \vee h_n \leftarrow b_1, b_2, \ldots, b_m$$

where h_i are logical atoms and b_i are logical literals, i.e., an atomic formula p or its negation $\neg p$. If p is an atomic formula, then p is its *positive* literal, $\neg p$ its *negative* literal and these two literals are said to be the *complement* of each other. If S is a set of literals, I denote the set formed by taking the complement of each literal in S by $\neg S$. The disjunction $h_1 \vee h_2 \vee \ldots \vee h_n$ is called the *head* of the clause and the conjunction $b_1 \wedge b_2 \wedge \ldots \wedge b_m$ is the called the *body*. Let us define the two functions $head(C)$ and $body(C)$ that, given a disjunctive clause C, return respectively the head and the body of C. In some cases, I will use the functions $head(C)$ and $body(C)$ to denote the set of the atoms in the head or of the literals of the body respectively. The meaning of $head(C)$ and $body(C)$ will be clear from the context. I indicate with $body(C)^+$ the set of positive literals of $body(C)$ and with $body(C)^-$ the set of its negative literals.

A term (clause) is ground if it does not contain variables. The Herbrand universe $H(P)$ of a disjunctive logic program P is the set of all the ground terms that can be constructed with the constant and function symbols appearing in P. The Herbrand base $H_B(P)$ of a disjunctive logic program P is the set of all the atoms constructed with the predicates appearing in P and the terms of $H(P)$. The grounding of a program P is the set of all the ground clauses obtained by picking a clause C from P and by substituting constants from $H(P)$ to each variable. If P contains function symbols then $H(P)$ and $H_B(P)$ are infinite but countable, otherwise they are finite. If P contains functions symbols and variables then its grounding is infinite but countable, otherwise it is finite. A Herbrand interpretation is a subset of $H_B(P)$. Let us denote the set of all Herbrand interpretations by $\mathcal{I}_P$.

Let us now discuss how to ascertain the truth of formulas in an interpretation. A ground atom a is true in an interpretation I iff $a \in I$. A ground negative literal $\neg a$ is true in an interpretation I iff $a \notin I$. A conjunction of ground literals from a set B is true iff $B^+ \subseteq I$ and $\neg B^- \cap I = \emptyset$, where B^+ (B^-) is the set of positive (negative) literals of B. A disjunction of ground atoms from a set D is true iff

$D \cap I \neq \emptyset$. A disjunctive non ground clause C is true in an interpretation I iff for all grounding substitution θ of C: $body(C)^+\theta \subset I \ \wedge \neg body(C)^-\theta \cap I = \emptyset \rightarrow head(C)\theta \cap I \neq \emptyset$. The truth of a ground clause is a simple case of this.

As was observed by [8], the truth of a range restricted clause C in a finite interpretation I can be tested by running the query $? - body(C), not\ head(C)$ on a database containing I. If the query succeeds C is false in I. If the query fails C is true in I.

A Logic Program with Annotated Disjunctions consists of a set of formulas of the form

$$(h_1 : p_1) \vee (h_2 : p_2) \vee \ldots \vee (h_n : p_n) \leftarrow b_1, b_2, \ldots b_m$$

called *annotated disjunctive clauses*. In such a clause the h_i are logical atoms, the b_i are logical literals and the p_i are real numbers in the interval $[0, 1]$ such that $\sum_{i=1}^{n} p_i = 1$. For a clause C of the form above, I define $head(C)$ as the set $\{(h_i : p_i) | 1 \leq i \leq n\}$ and $body(C)$ as the set $\{b_i | 1 \leq i \leq m\}$. If $head(C)$ contains a single element $(a : 1)$ I will simply denote the head as a. I denote with $\mathcal{P}_\mathcal{G}$ the set of all finite ground LPADs.

Example 1. Let us see an example of an LPAD taken from [38].

$$\begin{aligned}
(heads(Coin) : 0.5) \vee (tails(Coin) : 0.5) &\leftarrow toss(Coin), \neg biased(Coin).\\
(heads(Coin) : 0.6) \vee (tails(Coin) : 0.4) &\leftarrow toss(Coin), biased(Coin).\\
(fair(coin) : 0.9) \vee (biased(coin) : 0.1).&\\
toss(coin).&
\end{aligned}$$

The program can be read in the following way: if I toss a coin that is not biased, it has probability 50% of landing heads and probability 50% of landing tails. If I toss a coin that is biased, it has probability 60% of landing heads and probability 40% of landing tails. The coin *coin* is tossed and is fair with probability 90% and baised with probability 10%.

Example 2. Let us see another example of an LPAD taken from [4].

$$\begin{aligned}
father(f, s). \quad & mother(m, s).\\
cg(f, 1, p). \quad & cg(f, 2, w).\\
cg(m, 1, w). \quad & cg(m, 2, w).\\
(cg(X, 1, A) : 0.5) \vee (cg(X, 1, B) : 0.5) &\leftarrow father(Y, X), cg(Y, 1, A), cg(Y, 2, B).\\
(cg(X, 2, A) : 0.5) \vee (cg(X, 2, B) : 0.5) &\leftarrow mother(Y, X), cg(Y, 1, A), cg(Y, 2, B).\\
color(X, purple) &\leftarrow cg(X, _N, p).\\
color(X, white) &\leftarrow cg(X, 1, w), cg(X, 2, w).
\end{aligned}$$

This program encodes the Mendelian rules of inheritance of the color of pea plants. The color of a pea plant is determined by a gene that exists in two forms

(alleles), p and w, that stand for purple and white. Each plant has two alleles for the color gene that reside on a couple of chromosomes. *cg(X,N,A)* indicates that plant X has allele A on chromosome N. The facts of the program express that s is the offspring of f and m and that the alleles of f are pw and of m are ww. The disjunctive rules encode the fact that an offspring inherits the allele on chromosome 1 from the father and the allele on chromosome 2 from the mother. In particular, each allele of the parent has a probability of 50% of being transmitted. The definite clauses for *color* express the fact that the color of a plant is purple if at least one of the alleles is p, i.e., that the p allele is dominant.

2.2 Semantics of LPADs

The semantics of an LPAD was given in [38] for finite ground programs, i.e., programs in $\mathcal{P}_\mathcal{G}$. A non-ground program can be assigned a semantics only if its grounding is finite, i.e., if it does not contain function symbols. The semantics is given in this case in terms of its grounding. Note that a finite ground program can have an infinite Herbrand base in the case where it contains function symbols. Let us denote with $A_B(P)$ the set of all the atoms appearing in a program $P \in \mathcal{P}_\mathcal{G}$. $A_B(P)$, differently from $H_B(P)$, is always finite.

Each ground annotated disjunctive clause represents a probabilistic choice between a number of ground non-disjunctive clauses. By choosing a head atom for each ground clause of an LPAD I get a normal logic program called an *instance* of the LPAD. For example, the LPAD in example 1 has the following grounding

$$\begin{aligned}(heads(coin):0.5) \vee (tails(coin):0.5) &\leftarrow toss(coin), \neg biased(coin).\\ (heads(coin):0.6) \vee (tails(coin):0.4) &\leftarrow toss(coin), biased(coin).\\ (fair(coin):0.9) \vee (biased(coin):0.1).&\\ toss(coin).&\end{aligned}$$

since there is only the constant *coin* in the Herbrand universe of the program.

The LPAD has thus $2 \cdot 2 \cdot 2 \cdot 1 = 8$ possible instances one of which is (formula 1)

$$\begin{aligned}heads(coin) &\leftarrow toss(coin), \neg biased(coin).\\ heads(coin) &\leftarrow toss(coin), biased(coin).\\ fair(coin).&\\ toss(coin).&\end{aligned} \tag{1}$$

A probability is assigned to every instance by assuming independence between the choices made for each clause. Therefore, the probability of the instance above is $0.5 \cdot 0.6 \cdot 0.9 \cdot 1 = 0.27$.

An instance is identified by means of a selection function.

Definition 1 (Selection function [38]). *Let P be a program in $\mathcal{P}_\mathcal{G}$. A* selection *σ is a function which selects one pair $(h : p)$ from each rule of P, i.e.*

$\sigma : P \rightarrow (H_B(P) \times [0,1])$ *such that, for each* R *in* P, $\sigma(R) \in head(R)$. *For each rule* R, *I denote the selected atom* h *by* $\sigma_{atom}(R)$ *and the selected probability* p *by* $\sigma_{prob}(R)$. *Furthermore, I denote the set of all selections* σ *by* $\mathcal{S}_P$.

Let me now give the formal definition of an instance.

Definition 2 (Instance [38]). *Let* P *be a program in* $\mathcal{P}_\mathcal{G}$ *and* σ *a selection in* $\mathcal{S}_P$. *The* instance P_σ *chosen by* σ *is obtained by keeping only the atom selected for* R *in the head of each rule* $R \in P$, *i.e.* $P_\sigma = \{ \text{“}\sigma_{atom}(R) \leftarrow body(R)\text{”} | R \in P \}$.

I now assign a probability to a selection function σ and therefore also to the associated program P_σ.

Definition 3 (Probability of a selection [38]). *Let* P *be a program in* $\mathcal{P}_\mathcal{G}$. *The* probability of a selection σ *in* $\mathcal{S}_P$ *is the product of the probability of the individual choices made by that selection, i.e.*

$$C_\sigma = \prod_{R \in P} \sigma_{prob}(R)$$

The semantics of the instances of an LPAD can be given by any of the semantics defined for normal logic programs (e.g. Clark's completion [6], Fitting semantics [11], stable models [13], well founded semantics [36]). In [38] the authors have considered only the well founded semantics, the most skeptical one. Since in LPAD the uncertainty is modeled by means of the annotated disjunctions, the instances of an LPAD should contain no uncertainty, i.e. they should have a single two-valued model. Therefore, given an instance P_σ, its semantics is given by its well founded model $WFM(P_\sigma)$ and I require that it is two-valued.

Definition 4 (Sound LPAD [38]). $P \in \mathcal{P}_\mathcal{G}$ *is called* sound *iff for each selection* σ *in* $\mathcal{S}_P$, *the well founded model* $WFM(P_\sigma)$ *of the program* P_σ *chosen by* σ *is two-valued.*

For example, if the LPAD is acyclic (meaning that all its instances are acyclic [1]) then the LPAD is sound. I denote with $P_\sigma \models_{WFM} F$ the fact that the formula F is true in the well founded model of P_σ.

Let me note that, in the case a clause of a non ground LPAD generates two or more ground clauses, a selection can select different heads for the different groundings. For example, consider the following LPAD (formula 2)

$$\begin{aligned} (heads(Coin) : 0.5) \vee (tails(Coin) : 0.5) &\leftarrow toss(Coin), \neg biased(Coin). \\ (heads(Coin) : 0.6) \vee (tails(Coin) : 0.4) &\leftarrow toss(Coin), biased(Coin). \\ (fair(Coin) : 0.9) \vee (biased(Coin) : 0.1). & \\ toss(c1). & \\ toss(c2). & \end{aligned} \quad (2)$$

An instance of this LPAD is for example:

$$heads(c1) \leftarrow toss(c1), \neg biased(c1).$$

$$
\begin{aligned}
tails(c2) &\leftarrow toss(c2), \neg biased(c2).\\
tails(c1) &\leftarrow toss(c1), biased(c1).\\
heads(c2) &\leftarrow toss(c2), biased(c2).\\
fair(c1).&\\
biased(c2).&\\
toss(c1).&\\
toss(c2).&
\end{aligned}
$$

I now define the probability of interpretations.

Definition 5 (Probability of an interpretation [38]). *Let P be a sound program in $\mathcal{P}_\mathcal{G}$. For each of its interpretations I in $\mathcal{I}_P$, the probability $\pi^*_P(I)$ assigned by P to I is the sum of the probabilities of all selections which lead to I, i.e. with $S(I)$ being the set of all the selections σ for which $WFM(P_\sigma) = I$:*

$$\pi^*_P(I) = \sum_{\sigma \in S(I)} C_\sigma$$

For example, consider the interpretation $\{toss(coin), fair(coin), heads(coin)\}$. This interpretation is the well founded model of two instances of the LPAD of example 1, one is the instance represented by formula 1 and the other is the instance:

$$
\begin{aligned}
heads(coin) &\leftarrow toss(coin), \neg biased(coin).\\
tails(coin) &\leftarrow toss(coin), biased(coin).\\
fair(coin).&\\
toss(coin).&
\end{aligned}
$$

The probability of this latter instance is $0.5 \cdot 0.4 \cdot 0.9 \cdot 1 = 0.18$. Therefore, the probability of the interpretation above is $0.5 \cdot 0.4 \cdot 0.9 \cdot 1 + 0.5 \cdot 0.6 \cdot 0.9 \cdot 1 = 0.5 \cdot (0.4 + 0.6) \cdot 0.9 \cdot 1 = 0.45$.

Let me now introduce the definition of the probability of a formula.

Definition 6 (Probability of a formula [38]). *Let P be a sound program in $\mathcal{P}_\mathcal{G}$. For each formula ϕ, the probability $\pi^*_P(\phi)$ of ϕ according to P is the sum of the probability of all interpretations in which ϕ holds, i.e.*

$$\pi^*_P(\phi) = \sum_{I \in \mathcal{I}^\phi_P} \pi^*_P(I)$$

with $\mathcal{I}^\phi_P = \{I \in \mathcal{I}_P \ I \models \phi\}$.

In the case of the example 1, the probability of the formula $heads(coin)$ is given by the sum of the probabilities of the interpretations where $heads(coin)$ is true.

The interpretations where $heads(coin)$ is true that have a non-zero probability are:

$$I_1 = \{toss(coin), fair(coin), heads(coin)\}$$
$$I_2 = \{toss(coin), biased(coin), heads(coin)\}$$

where $\pi^*_P(I_1) = 0.45$ and $\pi^*_P(I_2) = (0.5+0.5)\cdot 0.6 \cdot 0.1 = 0.6 \cdot 0.1 \cdot 1 = 0.06$. Thus $\pi^*_P(heads(coin)) = 0.51$.

I can also define the conditional probability of a formula given another one.

Definition 7 (Conditional probability of a formula). *Let P be a sound LPAD in $\mathcal{P}_\mathcal{G}$. Given two formulas ϕ and ψ, the conditional probability of ϕ given ψ according to P is indicated by $\pi^*_P(\phi|\psi)$.*

According to Bayes theorem, I have

$$\pi^*_P(\phi|\psi) = \frac{\pi^*_P(\phi \wedge \psi)}{\pi^*_P(\psi)}$$

2.3 Well Founded Semantics

A normal logic program is a disjunctive logic program where each clause head contains a single atom, i.e., it is a logic program containing clauses C of the following form:

$$h \leftarrow b_1, b_2, \ldots, b_n$$

where h is an atom and each b_i is a literal. The literals b_i are called *subgoals* of C.

A literal q is *inconsistent* with a set of literals S iff $q \in \neg S$. Sets of literals S and R are *inconsistent* iff $R \cap \neg S \neq \emptyset$. A set of literals is *inconsistent* iff it is inconsistent with itself, otherwise it is *consistent*.

Given a normal logic program P, a *partial interpretation* I is a consistent set of literals whose atoms are in $H_B(P)$. A *total interpretation* is a partial interpretation that contains every atom of $H_B(P)$ or its negation. A total interpretation I corresponds to the Herbrand interpretation $I \cap H_B(P)$. I say a ground literal is *true in a partial interpretation* I when it is in I and it is *false in a partial interpretation* I when its complement is in I. I say that a ground conjunction of literals B is *true in a partial interpretation* I when $B \subseteq I$ and I say B is *false in a partial interpretation* I when $\neg B \cap I \neq \emptyset$. Note that if the partial interpretation is not total, then the fact that B is not true does not imply that it is false.

The definition of the well founded semantics was given in [36].

Definition 8 (Unfounded set [36]). *Let a normal program P, its associated Herbrand base $H_B(P)$ and a partial interpretation I be given. I say $A \subseteq H_B(P)$ is an* unfounded set *(*of *P)* with respect to *I if each atom $p \in A$ satisfies the following condition: For each instantiated clause C of P whose head is p, (at least) one of the following holds:*

1. *Some (positive or negative) subgoal q of C are false in I.*
2. *Some positive subgoal of C occurs in A.*

In other words: for each atom p of A, for each instantiated clause $p \leftarrow E$ of P, the following holds:

1. $\neg E \cap I \neq \emptyset$ *or*
2. $E \cap A \neq \emptyset$.

The union of arbitrary unfounded sets is an unfounded set. The *greatest unfounded set (of P) with respect to I*, denoted $U_P(I)$ is the union of all sets that are unfounded of P with respect to I.

A *transformation* is a function from a set of literals to a set of literals. A transformation T is *monotonic* iff $T(I) \subseteq T(J)$ whenever $I \subseteq J$.

Definition 9 (T_P, U_P and W_P transformations [36]). *Transformations T_P, U_P and W_P are defined as follows:*

- *$p \in T_P(I)$ iff there is some instantiated clause C of P such that C has head p and each subgoal in the body of C is true in I.*
- *$U_P(I)$ is the greatest unfounded set of P with respect to I, as in definition 8.*
- *$W_P(I) = T_P(I) \cup \neg U_P(I)$.*

Lemma 1 ([36]). *T_P, U_P and W_P are monotonic transformations.*

Definition 10 (I_α sets [36]). *Let α range over all countable ordinals. The sets I_α and I^∞, whose elements are literals whose atoms are in $H_B(P)$, are defined recursively by:*

1. *For limit ordinal α,*
$$I_\alpha = \bigcup_{\beta<\alpha} I_\beta$$
Note that 0 is a limit ordinal, and $I_0 = \emptyset$.
2. *For successor ordinal $\alpha = \gamma + 1$,*
$$I_{\gamma+1} = W_P(I_\gamma)$$
3. *Finally, define*
$$I^\infty = \bigcup_{\alpha} I_\alpha$$

For any literal p in I^∞, I define the stage *of p the least ordinal α such that $p \in I_\alpha$. For any conjunction of literals B I define the* stage *of B the least ordinal α such that $p \in I_\alpha$ for every literal p of B.*

Lemma 2 ([36]). *I_α as defined in Definition 10 is a monotonic sequence of partial interpretations (i.e., is consistent and $I_\beta \subseteq I_\alpha$ if $\beta < \alpha$)*

Definition 11 (Well founded semantics [36]). *The well founded semantics of a program P is the least fixed point of W_P, or the limit I^∞ described above. If I^∞ is a total interpretation, then I call it the* well-founded model*, otherwise I call it the* well-founded partial model*. In the first case I write $WFM(P) = I^\infty \cap H_B(P)$, i.e., $WFM(P)$ is the Herbrand interpretation corresponding to the total interpretation I^∞.*

Thus a program P is sound iff I^∞ is a total interpretation. I now define the dependency graph $LD(P)$ of a program P. The nodes of $LD(P)$ are the atoms of $H_B(P)$. The arcs are labeled either with $+$ or $-$ and are called respectively positive and negative arcs. There is a positive arc from node b to node a if there exists a clause in the grounding of P such that a is the head and b is in the body. There is a negative arc from b to a if there exists a clause in the grounding of P such that a is the head and $\neg b$ is in the body.

An atom a *depends positively* on an atom b if there is a path from b to a in the dependency graph containing an even number (zero included) of negative arcs. I also say that a *depends* on the literal b. An atom a *depends negatively* on an atom b if there is path from b to a in the dependency graph containing an odd number of negative arcs. I also say that a *depends* on the literal $\neg b$.

I will indicate with D_a the set of literals on which atom a depends. Note that it can be an inconsistent set. I indicate with D_a^- the set of negative literals of D_a.

If B is a conjunction of literals, I can define the dependencies of B by augmenting the local dependency graph with an extra node n_B, by drawing a positive arc to n_B from every atom a that appears positively in B and by drawing a negative arc to n_B from every atom a that appears negatively in B. Then the set of literals on which B *depends*, indicated by D_B, is defined as D_{n_B}.

A *negative path* from atom b to atom a is a path from b to a in the dependency graph that contains at least one negative arc. I will indicate with N_a the set of atoms that can reach a via a negative path.

I define N_B for a conjunction B in a way similar to the definition of D_B, i.e. $N_B = N_{n_B}$.

Lemma 3. *If a ground atom a of a program P belongs to D_X where X is either an atom or a conjunction of literals, then, for every clause $a \leftarrow B$ in the grounding of P, $B \subseteq D_X$. If a ground literal $\neg a$ belongs to D_X then, for every clause $a \leftarrow B$ in the grounding of P, $\neg B \subseteq D_X$. If a belongs to N_X then $N_X \supseteq N_a$.*

Proof. Obvious from the definition of dependency graph.

Lemma 4. *Let P be a program, let X be a ground atom of P or a conjunction of ground literals of P and let I be a partial interpretation of P. Then $A = U_P(I) \cap \neg D_X$ is an unfounded set of P with respect to I.*

Proof. I must verify that for each couple $(c, c \leftarrow E)$ such that $c \in A$ and $c \leftarrow E$ belongs to the grounding of P, it holds that $\neg E \cap I \neq \emptyset$ or $E \cap A \neq \emptyset$. Since A is a subset of $U_P(I)$, I must verify the conditions for those couples $(c, c \leftarrow E)$ for

which $\neg E \cap I = \emptyset$ and $E \cap U_P(I) \neq \emptyset$. Observe that, for lemma 3 and for the fact that $c \in \neg D_X$, then $\neg E \subseteq D_X$. Thus $E \subseteq \neg D_X$ and $E \cap A = E \cap \neg D_X \cap U_P(I) = E \cap U_P(I) \neq \emptyset$. So A is an unfounded set of P with respect to I.

Definition 12 (Locally stratified normal program [28]). *A normal program P is locally stratified iff all the atoms in $H_B(P)$ can be assigned assigned a countable ordinal* rank *such that, for every rule $h \leftarrow B$ in the grounding P' of P*

- *the rank of h is higher than the rank of every atom that appears negatively in B,*
- *the rank of h is higher or equal to the rank of every atom that appears positively in B.*

Theorem 1 ([36]). *If P is locally stratified, then it has a well founded model, i.e., I^∞ is a total interpretation.*

I extend the definitions of dependency graph and local stratification to disjunctive logic programs and thus to LPADs. The definition of dependency graph remains unaltered. The definition of local stratification is the natural extension of the one for normal programs.

Definition 13 (Locally stratified disjunctive program). *A disjunctive program P is locally stratified iff all of the atoms in $H_B(P)$ can be assigned a countable ordinal* rank *such that, for every rule $h_1 \vee \ldots \vee h_n \leftarrow B$ in the grounding P' of P, for every h_i*

- *the rank of h_i is greater than the rank of every atom that appears negatively in B,*
- *the rank of h_i is greater or equal to the rank of every atom that appears positively in B.*

Note that this definition of local stratification for disjunctive logic programs differs from the one in [28] because there the atoms in the head of a disjunctive clause were forced to have the same rank. I will now prove a lemma that will be useful in the following

Lemma 5. *If P is a finite ground program, then P is locally stratified iff there does not exist an atom $a \in H_B(P)$ such that $a \in N_a$.*

Proof. In one sense it is obvious. In the other sense, I want to show that if $\forall a \in H_B(P)\ a \notin N_a$ then P is locally stratified. I will do it by showing how to build a ranking that respects the definition of local stratification. Consider the set of atoms A_n defined recursively in the following way:

- $A_0 = \{a | a \in H_B(P) \wedge N_a = \emptyset\}$
- $A_n = \{a | a \in H_B(P) \setminus (\bigcup_{i=0}^{n-1} A_i) \wedge N_a \setminus (\bigcup_{i=0}^{n-1} A_i) = \emptyset\}$ for $n = 1, \ldots, \infty$

The ranking assigns rank n to all the atoms in A_n. Let us call $r(a)$ the rank of atom a, thus $r(a) = n$ iff $a \in A_n$. I will first prove that, if $\forall a \in H_B(P)$ $a \notin N_a$, then $\bigcup_{i=0}^{\infty} A_i = H_B(P)$. Then I will prove that the ranking satisfies the definition of local stratification.

If $\bigcup_{i=0}^{\infty} A_i \neq H_B(P)$ then consider the set $A = H_B(P) \setminus \bigcup_{i=0}^{\infty} A_i$. Since all the atoms $a \in H_B(P) \setminus A_B(P)$ belong to A_0 because $N_a = \emptyset$, then $A = A_B(P) \setminus \bigcup_{i=0}^{\infty} A_i$.

$a \in A$ is such that $\nexists n$ such that $a \in A_n$, i.e., $\forall n\ N_a \setminus (\bigcup_{i=1}^{n-1} A_i) \neq \emptyset$. This means that $N_a \setminus (\bigcup_{i=1}^{\infty} A_i) \neq \emptyset$. Since $N_a \subseteq A_B(P)$ then $N_a \setminus (\bigcup_{i=1}^{\infty} A_i) \subseteq A$. Thus $\forall a \in A\ \exists b \in A$ such that $b \in N_a$.

If $A \neq \emptyset$, since $\forall a \in A,\ a \notin N_a$, it is possible to order the elements a_i of A so that $a_i \in N_{a_{i+1}}$. But A is finite, i.e. $A = \{a_i, \ldots, a_m\}$. Then a_m must be such that $a_m \in N_{a_j}$ with $j \leq m$. Since $N_{a_{i+1}} \supseteq N_{a_i}$, then $N_{a_m} \supseteq N_{a_j}$, then $a_m \in N_{a_m}$ against the hypothesis, so A is empty.

Let us define P_n as the program containing all the clauses of P that contain an atom in the head belonging to $\bigcup_{i=0}^{n} A_i$. I will prove by induction that the ranking satisfies the definition of local stratification for every P_n thus also for $P = \bigcup_{n=0}^{\infty} P_n$.

Consider a clause $h_1 \vee \ldots \vee h_m \leftarrow b_1, \ldots, b_p, \neg c_1, \ldots, \neg c_r$ from P_0. In it $r = 0$ because it exists an a such that $h_a \in A_0$ thus $N_{h_a} = \emptyset$. Moreover, $\forall j\ b_j \in A_0$ because $\forall j\ N_{h_a} \supseteq N_{b_j}$. Since $\forall j\ r(b_j) = 0$ and $\forall i\ r(h_i) \geq 0$, the ranking satisfies the constraints for local stratification for P_0.

Now consider a program P_n. Given that the ranking satisfies the constraints for local stratification for P_{n-1}, I must prove that the constraints are satisfied for the clauses in $P_n \setminus P_{n-1}$. Let $h_1 \vee \ldots \vee h_m \leftarrow b_1, \ldots, b_p, \neg c_1, \ldots, \neg c_r$ be a clause from $P_n \setminus P_{n-1}$. For the definition of P_n, $\forall i\ \exists s\ h_i \in A_s$ with $s \geq n$ thus $\forall i\ r(h_i) \geq n$. Moreover, $\exists a\ h_a \in A_n$. This means that $N_{h_a} \setminus \bigcup_{l=0}^{n-1} A_l = \emptyset$ and, since $\forall j\ N_{h_a} \supseteq N_{b_j}$ and $\forall k\ N_{h_a} \supseteq N_{c_k}$, then $\forall j\ N_{b_j} \setminus \bigcup_{l=0}^{n-1} A_l = \emptyset$ and $\forall k\ N_{c_k} \setminus \bigcup_{l=0}^{n-1} A_l = \emptyset$ thus $\forall j\ \exists t\ b_j \in A_t$ with $t \leq n$ and $\forall k\ \exists s\ c_k \in A_s$ with $s \leq n$.

I will prove that in reality $s < n$. In fact if $s = n$ then $c_k \in A_n$ and, since $c_k \in N_{h_a}$, $c_k \in N_{h_a} \setminus \bigcup_{l=0}^{n-1} A_l$ thus $h_a \notin A_n$ against what I have assumed. Thus $\forall i, j\ r(h_i) \geq r(b_j)$ and $\forall i, k\ r(h_i) > r(c_k)$.

Lemma 6. *A finite ground disjunctive program P is locally stratified iff all its instances are locally stratified.*

Proof. An instance of P imposes on the rank a subset of the constraints that are imposed by P so if P is locally stratified then the instance is as well.

In the other direction, I will prove that if P is not locally stratified, then it exists an instance that is not locally stratified. By lemma 5 it exists an atom a such that $a \in N_a$. This means that it exists a negative path from a to a. Then consider the instance obtained by selecting the atoms on the path from the heads of the clauses giving rise to the arcs on the path and selecting any head from the other clauses. Such an instance is not locally stratified because in it $a \in N_a$.

Lemma 7. *Let P be a locally stratified normal program, let P' be its grounding and let Q be a (possibly countably infinite) subset of P'. Then Q is locally stratified.*

Proof. If there exists a ranking for P that satisfied the conditions of local stratification, the same ranking satisfies the conditions for Q because the conditions of Q are a subset of those for P.

I now present a lemma that will be used in the proof of theorem 5.

Lemma 8. *Let P, P^1 and P^2 be finite ground normal logic programs, with $P^1 = P \cup \{h_1 \leftarrow B\}$, $P^2 = P \cup \{h_2 \leftarrow B\}$ where h_1 and h_2 are ground atoms and B is a conjunction of ground literals. Moreover, let P^1 and P^2 be locally stratified. If $WFM(P^1) \models B$ then $WFM(P^2) \models B$.*

Proof. Let us first define D^1_B (D^2_B) as the set of literals on which B depends on in P^1 (P^2). Let us also consider the set D_B of literals on which B depends on in P. I will prove that $D^1_B = D^2_B = D_B$. Let us consider P^1. Consider the local dependency graph of P^1 augmented with the node n_B and with the arcs to it. It contains all the edges of the local dependency graph of P plus a positive edge to h_1 from every atom appearing in a positive literal of B and a negative edge to h_1 from every atom appearing in a negative literal of B.

Since P^1 is locally stratified, by lemma 5 $h_1 \notin N^1_B$ (N^1_B is the set of atoms that can reach n_B in P^1 via a negative path) and the paths from h_1 to every atom of B must contain zero negative arcs.

Thus an atom can reach n_B via a path in P containing an even number of negative arcs iff it can reach n_B via a path in P^1 containing an even number of negative arcs. So $D^1_B = D_B$.

The same reasoning can be applied for P^2. So $D^1_B = D^2_B = D_B$.

Moreover, iff $\neg h_1 \in D^1_B$, then $h_1 \in N^1_B$ and P_1 would not be locally stratified. Similarly for $\neg h_2$ and D^2_B. So $\neg h_1 \notin D_B$ and $\neg h_2 \notin D_B$.

Let I_α be the set defined in the well founded semantics for P. Similarly for I^1_α and I^2_α for P^1 and P^2 respectively.

Let ρ be the stage of B in P^1. I will prove that, for any ordinal $\alpha \leq \rho$, $I^1_\alpha \cap D_B = I_\alpha \cap D_B = I^2_\alpha \cap D_B$.

Since P^1 is finite, then ρ is finite as well, so I can prove the above statement using the principle of mathematical induction.

I first prove that for any ordinal $\alpha \leq \rho$, $I^1_\alpha \cap D_B = I_\alpha \cap D_B$. For $\alpha = 0$ the statement is true because $I^1_\alpha = I_\alpha = \emptyset$. For $\alpha \neq 0$, let $\delta = \alpha - 1$. Thus $I^1_\alpha = W_{P^1}(I^1_\delta)$ and $I_\alpha = W_P(I_\delta)$.

Consider a literal g belonging to $I^1_\alpha \cap D_B$. I will distinguish two cases: that g is positive and that g is negative.

If g is positive (i.e. $g = a$) then there exists a ground clause $a \leftarrow E \in P^1$ such that $E \subseteq I^1_\delta$. By lemma 3 $E \subseteq D_B$ thus $E \subseteq I^1_\delta \cap D_B$. For the inductive hypothesis $E \subseteq I_\delta \cap D_B$. Moreover, $a \leftarrow E$ can not be $h_1 \leftarrow B$ since B is not true in I^1_δ because $\delta < \rho$. Therefore $a \leftarrow E$ is also in P. Thus $a \in T_P(I_\delta)$ and $g \in I_\alpha \cap D_B$.

If g is negative (i.e. $g = \neg b$) then it belongs to $(\neg U_{P^1}(I^1_\delta)) \cap D_B$. By lemma 4 the set $A = U_{P^1}(I^1_\delta) \cap \neg D_B$ is unfounded of P^1 with respect to I^1_δ. From the definition of unfounded set, for every atom $c \in A$ and for each ground rule of P^1 of the form $c \leftarrow E$ it holds that $\neg E \cap I^1_\delta \neq \emptyset$ or $E \cap A \neq \emptyset$. I will now prove that A is also an unfounded set of P with respect to I_δ. In fact, for each atom $c \in A$, the set of ground rules of P of the form $c \leftarrow E$ is the same as that of P^1 since D_B does not contain $\neg h_1$. For each such rule, the conditions holding for P^1 also hold for P. The second condition is in fact the same. As regards the first condition, from lemma 3, $\neg E \subseteq D_B$. From $\neg E \cap I^1_\delta \neq \emptyset$ I have $\neg E \cap I^1_\delta \cap D_B \neq \emptyset$ and $\neg E \cap I_\delta \cap D_B \neq \emptyset$ for the inductive hypothesis. Thus A is also an unfounded set of P with respect to I_δ. Since $\neg g \in A$ then $\neg g \in U_P(I_\delta)$ and thus $g \in I_\alpha \cap D_B$.

Analogously I can prove that every literal g belonging to $I_\alpha \cap D_B$ also belongs to $I^1_\alpha \cap D_B$. So $I^1_\alpha \cap D_B = I_\alpha \cap D_B$ for every $\alpha \leq \rho$ is proved.

The proof that $I^2_\alpha \cap D_B = I_\alpha \cap D_B$ $\alpha \leq \rho$ can be conducted in a similar way, observing that if g is positive and the clause $g \leftarrow E$ is $h_2 \leftarrow B$ then B is true in $WFM(P^2)$ and the theorem is proved.

Thus $I^1_\rho \cap D_B = I^2_\rho \cap D_B$. Since $B \subseteq I^1_\rho$ and, given the definition of D_B, $B \subseteq D_B$, then $B \subseteq I^2_\rho$ and $WFM(P^2) \models B$.

The following two lemmas will be used in the proof of theorem 4. The principle of *transfinite induction* is as follows. Let $P(\alpha)$ be a property of ordinals. Suppose that, for all ordinals β, if $P(\gamma)$ holds for all $\gamma < \beta$, then $P(\beta)$ holds. Then $P(\alpha)$ holds for all ordinals α.

Lemma 9. *Consider the grounding P' of a sound normal logic program P and consider a (possibly countably infinite) set of ground clauses Z of P' such that every rule in Z has the body false in $WFM(P')$. Then $P' \setminus Z$ is sound and $WFM(P' \setminus Z) = WFM(P')$.*

Proof. Let us call P^1 the program P' and P^2 the program $P' \setminus Z$. Let I^1_α (I^2_α) be the I_α set of definition 10 for P^1 (P^2). Thus $I^1_\infty \cap H_B(P^1) = WFM(P^1)$.

I will first prove that $I^2_\alpha \supseteq I^1_\alpha \cap (H_B(P^2) \cup \neg H_B(P^2))$ for all α by using the principle of transfinite induction: I assume that, given an ordinal β, for all $\gamma < \beta$, $I^2_\gamma \supseteq I^1_\gamma \cap (H_B(P^2) \cup \neg H_B(P^2))$ and I prove that $I^2_\beta \supseteq I^1_\beta \cap (H_B(P^2) \cup \neg H_B(P^2))$.

I distinguish two cases: the one where β is a successor ordinal and the one where β is a limit ordinal.

If β is a successor ordinal, then there exist an ordinal δ such that $\delta + 1 = \beta$. Consider a literal g belonging to $I^1_\beta \cap (H_B(P^2) \cup \neg H_B(P^2))$, I will prove that it belongs to I^2_β.

If g is positive (i.e. $g = a$), then $a \in T_{P^1}(I^1_\delta)$. This means that there exist a ground clause $a \leftarrow E \in P^1$ such that $E \subseteq I^1_\delta$. But $a \leftarrow E \notin Z$ because otherwise $E \not\subseteq I^1_\infty$ and $E \not\subseteq I^1_\delta$ since $I^1_\delta \subseteq I^1_\infty$. Therefore $a \leftarrow E \in P^2$ and $E \subseteq H_B(P^2) \cup \neg H_B(P^2)$. Since $I^2_\delta \supseteq I^1_\delta \cap (H_B(P^2) \cup \neg H_B(P^2))$ then $E \subseteq I^2_\delta$ and $g \in I^2_\beta$.

If g is negative (i.e. $g = \neg a$), then $a \in U_{P^1}(I^1_\delta) \cap H_B(P^2)$. Let A be $U_{P^1}(I^1_\delta) \cap H_B(P^2)$, I will prove that A is an unfounded set of P^2 with respect to I^2_δ. For

all $c \in A$, for all $c \leftarrow E \in P^2$, I must prove that $\neg E \cap I^2_\delta \neq \emptyset$ or $E \cap A \neq \emptyset$. Since $c \in U_{P^1}(I^1_\delta)$ and $c \leftarrow E \in P^1$, then $\neg E \cap I^1_\delta \neq \emptyset$ or $E \cap U_{P^1}(I^1_\delta) \neq \emptyset$. The first condition implies that $\neg E \cap I^1_\delta \cap (H_B(P^2) \cup \neg H_B(P^2)) \neq \emptyset$ since $E \subseteq H_B(P^2) \cup \neg H_B(P^2)$. Thus for the inductive hypothesis $\neg E \cap I^2_\delta \neq \emptyset$. The second condition implies that $E \cap U_{P^1}(I^1_\delta) \cap H_B(P^2) \neq \emptyset$ since $E \subseteq H_B(P^2) \cup \neg H_B(P^2)$ and $U_{P^1}(I^1_\delta) \cap \neg H_B(P^2) = \emptyset$. Thus $E \cap A \neq \emptyset$. So A is an unfounded set of P^2 with respect to I^2_δ, $a \in U_{P^2}(I^2_\delta)$ and $g \in I^2_\beta$.

If β is a limit ordinal then

$$I^2_\beta = \cup_{\gamma<\beta} I^2_\gamma \text{ and } I^1_\beta = \cup_{\gamma<\beta} I^1_\gamma$$

For the inductive hypothesis $I^2_\gamma \supseteq I^1_\gamma \cap (H_B(P^2) \cup \neg H_B(P^2))$ for all $\gamma < \beta$, thus $I^2_\beta \supseteq I^1_\beta \cap (H_B(P^2) \cup \neg H_B(P^2))$.

So $I^2_\infty \supseteq I^1_\infty \cap (H_B(P^2) \cup \neg H_B(P^2))$. But I^1_∞ is two valued, i.e. every atom of $H_B(P^1)$ is present in positive or negative form in I^1_∞. Since $H_B(P^2) \subseteq H_B(P^1)$, then every atom of $H_B(P^2)$ is present in positive or negative form in I^1_∞. As a consequence, every atom of $H_B(P^2)$ is present in positive or negative form in $I^1_\infty \cap (H_B(P^2) \cup \neg H_B(P^2))$ and thus also in I^2_∞. So I^2_∞ is two valued (P^2 is sound) and $WFM(P^2) = I^2_\infty \cap H_B(P^2) = I^1_\infty \cap H_B(P^2)$. But $I^1_\infty \cap (H_B(P^1) \setminus H_B(P^2)) = \emptyset$, because all the clauses of Z have the body false in I^1_∞ and so no atom of $H_B(P^1) \setminus H_B(P^2)$ can be true in I^1_∞. So $WFM(P^1) = I^1_\infty \cap H_B(P^2)$ and $WFM(P^2) = WFM(P^1)$.

Lemma 10. *Consider the grounding P' of a sound normal logic program P and consider a (possibly countably infinite) set of ground clauses Z such that every clause in Z has the body false in the Herbrand interpretation $WFM(P')$ and such that $P' \cup Z$ is sound. Then $WFM(P' \cup Z) = WFM(P')$.*

Proof. Let us call P^1 the program $P' \cup Z$ and P^2 the program P'. Let I^1 be the total interpretation of P^1 corresponding to $WFM(P^2)$ (i.e. $I^1 = WFM(P^2) \cup \neg(H_B(P^1) \setminus WFM(P^2))$). Let I^2 be the total interpretation of P^2 corresponding to $WFM(P^2)$ (i.e. $I^2 = WFM(P^2) \cup \neg(H_B(P^2) \setminus WFM(P^2))$). The fact that the clauses of Z have the body false in the Herbrand interpretation $WFM(P^2)$ means that they have the body false in the total interpretation I^1, i.e., that $\forall a \leftarrow E \in Z$, $\neg E \cap I^1 \neq \emptyset$. Let I^1_+ (I^1_-) be the set of positive (negative) literals of I^1 and similarly for I^2_+ and I^2_-. Note that: $H_B(P^2) \subseteq H_B(P^1)$, $I^2_+ = I^1_+$, $I^2_- = I^1_- \cap \neg H_B(P^2)$, $I^1_- = I^2_- \cup \neg(H_B(P^1) \setminus H_B(P^2))$ and $I^2 \subseteq I^1$.

I will prove that I^1 is a fixed point of W_{P^1}, i.e., that $W_{P^1}(I^1) = I^1$, knowing that I^2 is a fixed point of W_{P^2}. I will do this by showing that $T_{P^1}(I^1) = T_{P^2}(I^2) = I^2_+ = I^1_+$ and that $U_{P^1}(I^1) = \neg I^1_-$.

As regards the first statement, let us consider an atom $g \in T_{P^1}(I^1)$. Then there exists a clause of P^1 of the form $g \leftarrow E$ such that E is true in I^1. This clause cannot be in Z since every clause of Z has the body false in I^1. Therefore this clause is also present in P^2. Thus the atoms of the literals of E are in $H_B(P^2)$ and $E \subseteq I^2$. Therefore $g \in T_{P^2}(I^2)$. Let us now consider an atom $g \in T_{P^2}(I^2)$. Then there exists a clause of P^2 of the form $g \leftarrow E$ such that $E \subseteq I^2$. This clause is also in P^1 and, since $I^2 \subseteq I^1$, E is true also in I^1. Thus $g \in T_{P^1}(I^1)$.

As regards the second statement, consider an atom $g \in U_{P^1}(I^1)$. Let us suppose that $g \in H_B(P^2)$. Then there exists an unfounded set $A \subseteq H_B(P^1)$ of P^1 with respect to I^1 such that $g \in A$. For every atom $a \in A$, for each clause $a \leftarrow E \in P^1$, $\neg E \cap I^1 \neq \emptyset$ or $E \cap A \neq \emptyset$. $A' = A \cap H_B(P^2)$ is an unfounded set of P^2 with respect to I^2. In fact, for every atom $a \in A'$ for each clause $a \leftarrow E \in P^2$, the atoms of the literals of E belong to $H_B(P^2)$ thus $\neg E \cap I^2 \neq \emptyset$ or $E \cap A' \neq \emptyset$. So $g \in U_{P^2}(I^2)$, $\neg g \in I^2$ and $\neg g \in I^1$ since $I^2 \subseteq I^1$. If $g \in H_B(P^1) \setminus H_B(P^2)$, then $\neg g \in I^1$.

Now consider an atom g such that $\neg g \in I^1$. If $g \in H_B(P^2)$ then $\neg g \in I^2$. So there exists an unfounded set A of P^2 with respect to I^2 such that $g \in A$. For every atom $a \in A$, for every clause $a \leftarrow E$ in P^2 it holds that $\neg E \cap I^2 \neq \emptyset$ or $E \cap A \neq \emptyset$. Then A is also an unfounded set of P^1. In fact, for the clauses $a \leftarrow E$ of P^2, $\neg E \cap I^1 \neq \emptyset$ or $E \cap A \neq \emptyset$ holds since $I^2 \subseteq I^1$. For the clauses $a \leftarrow E$ of Z, if holds that $\neg E \cap I^1 \neq \emptyset$. Thus $g \in U_{P^1}(I^1)$. If $g \in H_B(P^1) \setminus H_B(P^2)$ then $A' = H_B(P^1) \setminus H_B(P^2)$ is an unfounded set of P^1 with respect to I^1. In fact, for every atom $a \in A'$, every clause of the form $a \leftarrow E$ must belong to Z, so E is false in I^1. So $g \in U_{P^1}(I^1)$.

Due to proposition 5.1 in [22], the least fixed point of W_{P^1} is the intersection of all the fixed points of W_{P^1}. Since I^1 is a fixed point, then the least fixed point of W_{P^1} is a subset of I^1. If the least fixed point of W_{P^1} is a proper subset of I^1 it would not be total, contrary to the fact that, since P^1 is sound, the least fixed point of W_{P^1} is a total interpretation. Thus the least fixed point of W_{P^1} is equal to I^1. So $WFM(P^1) = I^1_+ = WFM(P^2)$.

Note that the fact that $P' \cup Z$ is sound is necessary for the lemma to hold. In fact consider the program $P' = \{a \leftarrow \neg b\}$ and the set $Z = \{b \leftarrow \neg a\}$. Then $WFM(P') = \{a\}$ and the body of the clause in Z is not true in $WFM(P')$ but $P' \cup Z$ has the well founded partial model $\emptyset$.

Let me now introduce some definitions and theorems that will be needed for proving the correctness of ALLPAD (theorem 6). I first introduce the Fitting semantics [12].

Definition 14. *Given a normal program P, N_P is defined as the transformation that, given a partial interpretation I, gives as $N_P(I)$ the set of atoms p such that, for every instantiated rule of P of the form $p \leftarrow B$, B is false in I, i.e. $\neg B \cap I \neq \emptyset$. Note that N_p is the portion of U_P produced by condition (1) of definition 8. The transformation F_P is defined in the following way: $F_P(I) = T_P(I) \cup \neg N_P(I)$.*

Definition 15. *Given a normal program P, its Fitting model is the least fixed point of the transformation F_P.*

Theorem 2 (Corollary 4.3 of [36]). *Given a normal program P, the Fitting model of P is a subset of I^∞.*

I now define a class of programs for which the Fitting and the well founded semantics coincide, the class of acyclic programs.

Definition 16 ([1]). *A normal program P is acyclic iff all of the atoms in $H_B(P)$ can be assigned a countable ordinal* rank *such that, for every rule $p \leftarrow B$ in the grounding P, the rank of p is greater than the rank of every atom on which a literal that appears in B is built.*

Theorem 3 (Theorem 11.1 in [2]). *If P is an acyclic program, then (1) $FM(P) = WFM(P)$ and (2) they are two valued.*

I will now give the definition of an acyclic LPAD.

Definition 17. *An LPAD $P \in \mathcal{P}_\mathcal{G}$ is acyclic iff all of the atoms in $H_B(P)$ can be assigned a countable ordinal* rank *such that, for every rule $h_1 : p_1 \vee \ldots \vee h_n : p_n \leftarrow B$ in the grounding P, every h_i has the same rank r and r is greater than the rank of every atom on which a literal that appears in B is built.*

3 Properties of LPADs

In [30] a definition and two theorems were given that will be useful in the following. I will report them here, together with a correction: in [30] it was not specified that the theorems hold only for ground clauses and for locally stratified programs. Moreover, I report also more detailed proofs for them.

The first theorem states that, under certain conditions, the probabilities of the head disjuncts of a ground rule can be computed from the probabilities of the interpretations. In particular, the probability of a disjunct h_i is given by the conditional probability of h_i given the body, i.e. by the sum of the probabilities of interpretations where the body of the clause and h_i are true divided by the sum of the probabilities of interpretations where the body is true.

The second theorem states that, given an interpretation I, under certain conditions, all the selections σ in the set $S(I)$ agree on all the rules in program with the body true in I and that the probability of I can be computed by multiplying the probabilities of the head disjuncts selected by a $\sigma \in S(I)$ for all the clauses with the body true.

Definition 18 (Mutually exclusive bodies). *Ground clauses $H_1 \leftarrow B_1$ and $H_2 \leftarrow B_2$ have* mutually exclusive bodies over a set of interpretations J *if, $\forall I \in J$, B_1 and B_2 are not both true in I.*

Theorem 4. *Consider a locally stratified LPAD P in $\mathcal{P}_\mathcal{G}$ and a clause $C \in P$ of the form*

$$C = h_1 : p_1 \vee h_2 : p_2 \vee \ldots h_m : p_m \leftarrow B.$$

Suppose you are given the function π_P^* and suppose that all the clauses of P that share an atom in the head with C have mutually exclusive bodies with C over the set of interpretations $J = \{I | \pi_P^*(I) > 0\}$. The probabilities p_i are given by the conditional probability of the head atoms given the body:

$$p_i = \pi_P^*(h_i|B)$$

Proof. Using Bayes theorem I get

$$p_i = \frac{\pi^*_P(h_i \wedge B)}{\pi^*_P(B)}$$

which gives

$$p_i = \frac{\sum_{I \in \mathcal{I}_P, I \models B, h_i} \pi^*_P(I)}{\sum_{I \in \mathcal{I}_P, I \models B} \pi^*_P(I)}$$

Let us first expand the numerator:

$$\sum_{I \in \mathcal{I}_P, I \models B, h_i} \pi^*_P(I) = \sum_{I \in J, I \models B, h_i} \sum_{\sigma \in S(I)} \prod_{R \in P} \sigma_{prob}(R)$$

A selection σ such that $WFM(P_\sigma) = I$ for an I such that $I \models B, h_i$ is a selection such that $P_\sigma \models_{WFM} B, h_i$. Therefore the above expression can be written as

$$\sum_{\sigma \in T} \prod_{R \in P} \sigma_{prob}(R)$$

where $T = \{\sigma | P_\sigma \models_{WFM} B, h_i\}$. Since clause C has a mutually exclusive body over the set of interpretations J with all the other clauses of P that contain h_i in the head, the truth of h_i in P_σ can be obtained only if $\sigma(C) = (h_i : p_i)$ for all $\sigma \in T$, therefore the numerator becomes

$$\sum_{\sigma \in T} \prod_{R \in P \setminus \{C\}} \sigma_{prob}(R)\sigma_{prob}(C) =$$
$$\sum_{\sigma \in T} \prod_{R \in P \setminus \{C\}} \sigma_{prob}(R)p_i =$$
$$p_i \sum_{\sigma \in T} \prod_{R \in P \setminus \{C\}} \sigma_{prob}(R)$$

Let us expand the denominator in a similar way

$$\sum_{I \in \mathcal{I}_P, I \models B} \pi^*_P(I) = \sum_{\sigma \in Q} \prod_{R \in P} \sigma_{prob}(R)$$

where $Q = \{\sigma | P_\sigma \models_{WFM} B\}$. Let us define Q_j as the set that contains all the selections σ that differ from a selection σ' from T only on clause C in the following way: $\sigma(C) = (h_j : p_j)$ while $\sigma'(C) = (h_i : p_i)$. Then $Q_j \cap Q_k = \emptyset$ for all $j, k = 1, \ldots m, j \neq k$.

I will show that $Q = Q_1 \cup \ldots \cup Q_m$. I start by showing that if $\sigma \in Q_j$ then $\sigma \in Q$. Let σ be a selection in Q_j and let σ' be the selection from T that differs from σ only on clause C. Thus $P_\sigma = (P_{\sigma'} \setminus \{h_i \leftarrow B\}) \cup \{h_j \leftarrow B\}$. Since $P_{\sigma'} \models_{WFM} B, h_i$ then $P_{\sigma'} \models_{WFM} B$. From lemma 8 and from the fact that P is locally stratified, $P_\sigma \models_{WFM} B$. Thus $\sigma \in Q$.

I now show that if $\sigma \in Q$ then there exists a j such that $\sigma \in Q_j$. Let σ be a selection in Q, then $P_\sigma \models_{WFM} B$. Let $(h_j : p_j)$ be the disjunct that σ selects

on C. Now consider a selection σ' that differs from σ only on clause C: $\sigma'(C) = (h_i : p_i)$. Then $P_{\sigma'} = (P_\sigma \setminus \{h_j \leftarrow B\}) \cup \{h_i \leftarrow B\}$. From lemma 8 and from the fact that P is locally stratified, $P_{\sigma'} \models_{WFM} B$. Moreover $P_{\sigma'} \models_{WFM} B, h_i$. Thus $\sigma' \in T$ and $\sigma \in Q_j$.

Therefore the denominator becomes

$$\sum_{j=1}^{m} \sum_{\sigma \in Q_j} \prod_{R \in P} \sigma_{prob}(R) =$$

$$\sum_{j=1}^{m} \sum_{\sigma \in Q_j} \prod_{R \in P \setminus \{C\}} \sigma_{prob}(R) p_j =$$

$$\sum_{j=1}^{m} p_j \sum_{\sigma \in Q_j} \prod_{R \in P \setminus \{C\}} \sigma_{prob}(R)$$

By the definition of Q_j the elements of T and Q_j are in one-to-one correspondence and

$$\sum_{\sigma \in T} \prod_{R \in P \setminus \{C\}} \sigma_{prob}(R) = \sum_{\sigma \in Q_j} \prod_{R \in P \setminus \{C\}} \sigma_{prob}(R)$$

for all $j = 1, \ldots, m$. Thus, the fraction becomes

$$\frac{p_i \sum_{\sigma \in T} \prod_{R \in P \setminus \{C\}} \sigma_{prob}(R)}{\left(\sum_{j=1}^{m} p_j\right) \sum_{\sigma \in T} \prod_{R \in P \setminus \{C\}} \sigma_{prob}(R)} = p_i$$

Theorem 5. *Consider an interpretation I and a locally stratified LPAD P in $\mathcal{P}_\mathcal{G}$ such that all the couples of clauses of P that share an atom in the head have mutually exclusive bodies with respect to the set of interpretations $\{I\}$. If $S(I) \neq \emptyset$ then all the selection $\sigma \in S(I)$ agree on the clauses of P with body true in I and*

$$\pi^*_P(I) = \prod_{R \in P, I \models body(R)} \sigma_{prob}(R) \tag{3}$$

where σ is any element of $S(I)$. If $S(I) = \emptyset$ then $\pi^*_P(I) = 0$.

Proof. If $S(I) = \emptyset$ then, by the definition of $\pi^*_P(I)$, $\pi^*_P(I) = 0$.

If $S(I) \neq \emptyset$, consider the set F of clauses of P with the body false in I. Since P is finite, F is also finite. Thus $F = \{R_1, R_2, \ldots, R_m\}$. Consider the following sequence of subprograms of P: $P^0 = P \setminus F$ and $P^n = P^{n-1} \cup \{R_n\}$ for $1 \leq n \leq m$. Note that, by lemma 7 and by the fact that P is locally stratified, every P^n is locally stratified.

Let $S_n(I)$ be the set of all selections σ such that $WFM(P^n_\sigma) = I$. I will first prove that $S_n(I) \neq \emptyset$ for all $n = 0, \ldots, m$. Consider a selection $\sigma \in S(I)$ and the program P_σ selected by it. Moreover, let P^n_σ be the program selected by it from P^n. $P^n_\sigma = P_\sigma \setminus T$ where T is a set of clauses with the body false in I and P_σ is locally stratified. For lemma 9 $WFM(P^n_\sigma) = I$ thus the restriction of σ to the clauses of P^n belongs to $S_n(I)$.

Now I will prove formula 3 by induction on the sequence of programs P^n.

Case $n = 0$. For each atom $a \in I$, there is only one clause C of P^0 that has it in the head for the assumption of mutual exclusion. Therefore, for a to be in I, every $\sigma \in S(I)$ must select atom a for clause C. Moreover, all the clauses of P^0 have the body true in I, therefore for each clause one atom in the head must be in I. Therefore there is a single σ in $S(I)$ and the theorem holds.

I assume that the theorem holds for P^{n-1} and I prove that the theorem holds for P^n. Suppose that R_n is

$$h_1 : p_1 \vee h_2 : p_2 \vee \ldots \vee h_m : p_m \leftarrow B$$

Moreover, let $S_n^i(I)$ be the set of all the selections σ of P^n such that they extend a selection σ' of $S_{n-1}(I)$ over clause R_n (i.e., $\forall R \in P^{n-1} : \sigma(R) = \sigma'(R)$) and $\sigma(R_n) = (h_i : p_i)$. Thus $S_n^i \cap S_n^j = \emptyset$ for all $i, j = 1, \ldots, m, i \neq j$.

I will show that $S_n(I) = S_n^1(I) \cup \ldots \cup S_n^m(I)$. Consider a selection $\sigma \in S_n(I)$: σ is such that $WFM(P_\sigma^n) = I$. Let us call $(h_i : p_i)$ the disjunct it selects from R_n. Consider the selection σ' obtained by restricting σ on the clauses of P^{n-1}. Since R_n has the body false in I and P^n is locally stratified, by lemma 9 then I is also the well founded model of $P_{\sigma'}^{n-1}$. Thus $\sigma \in S_n^i(I)$. Now consider a selection $\sigma \in S_n^i(I)$. By the definition of $S_n^i(I)$, σ is such that $WFM(P_\sigma^{n-1}) = I$. Since R_n has the body false in I and P^n is locally stratified by lemma 10 I is also the well founded model of P_σ^n. Thus $\sigma \in S_n(I)$.

I can write:

$$\begin{aligned}
\pi_{P^n}^*(I) &= \sum_{\sigma \in S_n(I)} \prod_{R \in P^n} \sigma_{prob}(R) = \\
&= \sum_{i=1}^{m} \sum_{\sigma \in S_n^i(I)} \prod_{R \in P^n} \sigma_{prob}(R) = \\
&= \sum_{i=1}^{m} \sum_{\sigma \in S_n^i(I)} \prod_{R \in P^{n-1}} \sigma_{prob}(R)\sigma_{prob}(R_n) = \\
&= \sum_{i=1}^{m} \sum_{\sigma \in S_n^i(I)} \prod_{R \in P^{n-1}} \sigma_{prob}(R)p_i = \\
&= \sum_{i=1}^{m} p_i \sum_{\sigma \in S_n^i(I)} \prod_{R \in P^{n-1}} \sigma_{prob}(R) =
\end{aligned}$$

Since there is a one to one correspondence between elements of $S_n^i(I)$ and elements of $S_{n-1}(I)$ and the corresponding selections agree on all the clauses in P^{n-1} then

$$\pi_{P^n}^*(I) = \sum_{i=1}^{m} p_i \sum_{\sigma \in S_{n-1}(I)} \prod_{R \in P^{n-1}} \sigma_{prob}(R) =$$

$$= \sum_{\sigma \in S_{n-1}(I)} \prod_{R \in P^{n-1}} \sigma_{prob}(R) \sum_{i=1}^{m} p_i =$$
$$= \sum_{\sigma \in S_{n-1}(I)} \prod_{R \in P^{n-1}} \sigma_{prob}(R)$$

which, for the hypothesis for $n-1$, becomes

$$\prod_{R \in P^{n-1}, I \models body(R)} \sigma_{prob}(R)$$

where σ is any element of $S_{n-1}(I)$. Since R_n has the body false in I,

$$\pi^*_{P^n}(I) = \prod_{R \in P^n, I \models body(R)} \sigma_{prob}(R) \tag{4}$$

I now have to prove that every element of $S_n(I)$ agrees with every element of $S_{n-1}(I)$ on the clauses with the body true in I, i.e., on the clauses of P^0. Since I proved that $S_n(I) = \bigcup_{i=1}^{n} S_n^i(I)$, $\sigma \in S_n(I)$ is such that it exists an i such that $\sigma \in S_n^i(I)$. By the way in which $S_n^i(I)$ was constructed, then σ agrees with a selection of $S_{n-1}(I)$ on all the clauses of P^{n-1}. The inductive hypothesis states that all the selections in $S_{n-1}(I)$ agree on all clauses of P^0, so all the selections of $S_n(I)$ agree on the clauses of P^0. So formula 4 holds with σ any element of $S_n(I)$ and the theorem holds.

The hypothesis of mutual exclusion of the bodies is fundamental for this theorem to hold. In fact, consider the following example:

$$P_2 = a : a_1 \vee b : b_1 \vee c : c_1.$$
$$a : a_2 \vee c : c_2 \vee d : d_2.$$
$$a : a_3 \vee b : b_3 \vee d : d_3.$$

Then $\pi^*_{P_2}(\{a, b, c\}) = a_1 c_2 b_3 + b_1 c_2 a_3 + c_1 a_2 b_3$.

4 Learning LPADs

I consider a learning problem of the following form [29]:
Given:

- a set E of examples that are couples $(I, \pi(I))$ where I is an interpretation, $\pi(I)$ is its associated probability and $\sum_{(I,\pi(I)) \in E} \pi(I) = 1$,
- a space of possible LPAD S (described by a language bias LB)

Find: an LPAD $P \in S$ such that $\forall (I, \pi(I)) \in E \;\; \pi^*_P(I) = \pi(I)$

Instead of a set of couples $(I, \pi(I))$, the input of the learning problem can be a multi set E' of interpretations. From this case I can obtain a learning problem of the form above by computing a probability for each interpretation in E'. The probability can be computed by dividing the number of occurrences of the interpretation by the total number of interpretations in E'.

I now describe LLPAD [29] and then ALLPAD.

4.1 LLPAD

LLPAD learns ground LPADs in four phases: the first consists in finding all the definite clauses that satisfy certain constraints, the second consists in finding all the disjunctive clauses that satisfy certain constraints, the third consists in annotating the head atoms of disjunctive clauses with probabilities and the fourth consists in solving a constraint satisfaction problem.

The first and second phases can be cast in the framework proposed by [34] in which the problem of descriptive ILP is seen as the problem of finding all the clauses in the language bias that satisfy a number of constraints. Exploiting the properties of constraints, the search in the space of clauses can be usefully pruned.

A constraint is *monotonic* if it is the case that when a clause does not satisfy it, none of its generalizations (in the θ-subsumption generalization order) satisfies it. A constraint is *anti-monotonic* if it is the case that when a clause does not satisfy it, none of its specializations satisfies it.

The first phase of LLPAD can be formulated in this way: find all the definite clauses that satisfy the following constraints:

D1 they have their body true in at least one interpretation
D2 they are satisfied in all the interpretations
D3 they are maximally general (there does not exist another clause that is strictly more general and that satisfies D1 and D2)

LLPAD searches the space of definite clauses by performing a complete depth-first and top-down search in the space of bodies for each ground atom allowed by the language bias in the head of clauses, exploiting constraint D3 and D1: as soon as a body is found such that the clause is satisfied in all interpretations, the clause is returned and the search along that branch is stopped; as soon as a body that is true is zero interpretations is found, the search along that branch is stopped, because constraint D1 is anti-monotonic.

The second phase can be formulated in this way: find all the disjunctive clauses that satisfy the following constraints:

C1 they have their body true in at least one interpretation
C2 they are satisfied in all the interpretations
C3 their atoms in the head are mutually exclusive over the set of interpretations where the body is true (i.e. no two head atoms are both true in an interpretation where the body is true)
C4 they have no head atom true in no interpretation where the body is true

LLPAD searches the space of disjunctive clauses by first searching depth-first and top-down for bodies true in at least one interpretation and then, for each such body, searching for a head satisfying the remaining constraints. When a body is found that is true in zero interpretations, the search along that branch is stopped (constraint C1 is anti-monotonic).

The system searches the clause space in a complete way and employs only bottom-up search in the space of heads, exploiting the monotonic constraint

C2 that requires the clause to be true in all the interpretations for pruning the search.

Definite clauses are searched separately because if I allow the search in the space of heads in the second phase to reach single atom heads I would return also non maximally general definite clauses. In fact, given a body B, an atom a and a literal b, if $a \leftarrow B$ satisfies constraints C1-4 and B, b is true in at least one interpretation, then both the clauses $a \leftarrow B$ and $a \leftarrow B, b$ will be returned, thus violating constraint D3.

The third phase is performed by using theorem 4: given a ground clause generated by the second phase, the probabilities of head atoms are given by the conditional probability of the head atoms given the body according to the distribution π.

In the fourth phase, LLPAD partitions the found disjunctive clauses in subsets that are solutions of the learning problem. This is done by assigning to each clause C_i found in the second phase a variable x_i that is 0 if the clause is absent from a solution P and is 1 if the clause is present. The system must ensure that the couples of clauses that share a literal in the head have mutually exclusive bodies over the set of interpretations E. This is achieved by testing, for each couple of clauses, if they share a literal in the head and, if so, if the intersections of the two sets of interpretations where their body is true is non-empty. LLPAD asserts the constraint $x_i + x_j \leq 1$ for all such couples of clauses (C_i, C_j). Finally P must assign the correct probability to each interpretation. For each interpretation I such that $(I, \pi(I)) \in E$, the system asserts the constraint:

$$\prod_{C_i \in SC(I)} p_i^{x_i} = \pi(I)$$

where $SC(I)$ is the set of all the found disjunctive clauses whose body is true in I and p_i is the probability of the single head of C_i that is true in I. This constraint is based on theorem 5.

Definite clauses are not considered in the constraints because they would contribute only with a factor 1^{x_j} that has no effect on the constraint for any value of x_j. Therefore, for each assignment of the other variables, x_j can be either 0 or 1. For this reason, all the found definite clauses are included in every solution and, therefore, I need only the most general definite clauses.

If I take the logarithm of both members I get the following linear constraint:

$$\sum_{C_i \in SC(I)} x_i \log p_i = \log \pi(I) \tag{5}$$

LLPAD can thus find the solutions of the learning problem by solving the above linear constraint satisfaction problem.

4.2 ALLPAD

ALLPAD learns ground LPADs in five phases. The first and the third are the same as those of LLPAD. The second and the fourth modify those of LLPAD and the fifth one is new.

Let me first consider the fourth phase: in it ALLPAD does not solve a constraint satisfaction problem but it solves an optimization problem. In fact, with real world problems, a perfect solution of the learning problem may not exist in the solution space, because the language bias is too restricted or because the data available is noisy. In these cases the constraint phase returns a failure. In order to solve this problem, I may enlarge the language bias, but this can lead to unacceptable run times, or look for an approximate solution. Thus the constraint problem is transformed into an optimization problem where the system tries to satisfy the interpretation constraints as much as possible. i.e., it tries to minimize the absolute value of the difference between the left and right members of the interpretation constraints (5).

However the absolute value function is not linear, therefore I have to transform the cost function so that I can use linear programming techniques. To this purpose, I introduce a "slack variable" named max and two "slack variables" s_I^+ and s_I^- for each interpretation I. These variables are real. Then each interpretation constraint of the form of equation (5) is substituted with two constraints of the form

$$\frac{\sum_{C_i \in SC(I)} x_i \log p_i}{\log \pi(I)} - 1 \leq s_I^+$$

$$1 - \frac{\sum_{C_i \in SC(I)} x_i \log p_i}{\log \pi(I)} \leq s_I^-$$

Moreover they must satisfy the following constraints for every I

$$s_I^+ \geq 0 \qquad s_I^+ \leq max \qquad s_I^- \geq 0 \qquad s_I^- \leq max$$

The cost function to be minimized can now be expresses as

$$0.5 \times max + 0.5 \times \frac{\sum_{(I,\pi(I)) \in E} (s_I^+ + s_I^-)}{|E|}$$

In this way I try to minimize both the maximum error and the average error, assigning them the same weight.

Since now both the constraints and the cost function are linear, I can use mixed-integer programming (MIP) techniques. If no perfect solution exist, a non zero optimum will be found.

However, the optimization problem, as the constraint problem, is NP-hard and thus solvable only for small instances. To overcome this problem, I exploit the possibility of setting a time limit offered by many MIP packages: at the deadline, the best admissible solution found up to that point is returned. An admissible solution in this case is one that respects all the mutual exclusion constraints. If no admissible solution has been found, the package returns an error. In this way, ALLPAD looks for the best solution given the available time.

To make sure that an admissible solution will be found within the time limits, one can reduce the dimension of the problem by reducing the number of clauses found in the second phase. Thus one may prefer the clauses that apply

to examples with a high probability, because they will give a large contribution to the probability of interpretations. For this reason the complete search in the space of bodies performed by LLAPD is given up for an incomplete search strategy, beam search. The heuristic to be used for ranking bodies is the sum of the probabilities of the interpretations where the body is true. This heuristic ensures that the clauses that apply only to a small number of improbable interpretations are discarded and the dimension of the optimization problem is reduced.

Moreover, in ALLPAD it is possible to set a limit on the number of nodes explored in the search for bodies, in order to further limit the number of generated clauses.

As regards the search in the space of heads, the system can employ either bottom-up search or top-down search at the user choice. When it searches bottom-up it exploits the monotonic constraint that requires the clause to be true in all the interpretations for pruning the search. When it searches top-down it exploits the anti-monotonic constraint that requires head atoms to be mutually exclusive. This second possibility was proposed in [34] where the authors present the system *Classic'cl*.

ALLPAD can also employ a third modality for finding heads that does not require search. If the head atoms in a clause templates in the language bias are mutually exclusive by construction, then a clause can be found in the following way: the system starts with a head containing all the possible atoms then, if the clause is satisfied in all interpretations, it removes the head atoms that are not true in at least one interpretation where the body is true and it returns the clause. This modality has been employed in the experiments presented in section 7.

ALLPAD uses complete search in the space of heads, because head search gives at most one clause as output and, with an incomplete search, it may fail in finding the clause satisfying the constraints, thus discarding a body that applies to many probable examples.

The algorithm for the second phase in pseudo-code is given in Figure 1. In it the function SearchHeads performs the search in the space of heads and it returns either a set containing a single clause satisfying the constraints or the empty set if no such clause exists.

The language biases and refinement operators used in the first and second phases are described in the next section.

In the fifth phase, the definite clauses not mutually exclusive with the selected disjunctive clauses are removed. To this purpose, for each definite clause, ALLPAD looks for the disjunctive clauses that share an atom in the head with the definite clause and checks if the bodies are both true in one of the input interpretations. If this is true, the definite clause is removed. In this way in the output program all clauses sharing an atom in the head have mutually exclusive bodies.

Fig. 1. Function SecondPhase

```
function SecondPhase(
    inputs : E : set of couples (I, π(I)),
        LB : a language bias expressed as a set of clause templates,
        d : dimension of the beam
        n : maximum number of explored nodes
    returns : C : a set of disjunctive clauses)

C := ∅
for each clause template T in LB
    let ρ_T be the downward refinement operator for
        bodies relative to clause template T
    Beam := [(true, Σ_{(I,π(I))∈E} π(I))]
        /* Beam is a list of couples ordered on the second argument */
    i := 0
    while Beam ≠ ∅ and i < n
        i := i + 1
        remove the first couple (Body, P) from Beam
        if Body is true in at least one interpretation (i.e. P > 0) then
            C := C∪SearchHeads(E, Body, LB)
            Ref:= ρ_T(Body)
            for each body R ∈Ref
                let ER be the set of couples (I, π(I)) of E such that
                    R is true in I
                PR := Σ_{(I,π(I))∈ER} π(I)
                insert (R, PR) in Beam in descending order of PR
            discard the elements of Beam after the d-th
return C
```

5 Correctness

In this section I will prove that ALLPAD is correct in the case in which a minimum of 0 is reached in the optimization phase, i.e., if the solution returned by ALLPAD has a cost of 0, then the solution assign to every input interpretations the given probability.

ALLPAD is not complete because it performs beam search in the space of bodies and because it does not find the optimum of the optimization problem.

Theorem 6 (Correctness of ALLPAD). *Given a learning problem LP such that the language bias allows only acyclic programs, if the optimization phase of ALLPAD finds a solution Q with cost equal to 0, then the solution satisfies the conditions of LP.*

Proof (of theorem 4). We have to prove that, for all $(I, \pi(I)) \in E$, $S(I) \neq \emptyset$. In this case in fact theorem 5 can be applied.

To this purpose, given an interpretation I such that $(I, \pi(I)) \in E$, let Q_σ be the normal program obtained by selecting the atom that is true in I from the rules of the solution whose body is true in I and any atom from the other rules. We have to prove that $WFM(Q_\sigma) = I$. Note that, by the fact that the language bias allows only acyclic programs, then $WFM(Q_\sigma)$ exists and is two valued for every σ

Let $I = \{l_1, \ldots, l_n\}$ be a total interpretation where l_i is a literal for $i = 1, \ldots, n$. Let I^+ (I^-) be the set of positive (negative) literals of I. I^+ is the corresponding Herbrand interpretation. The probability of I can be expressed as

$$\pi(I) = \pi(l_1 \wedge \ldots \wedge l_n)$$

since the formula $l_1 \wedge \ldots \wedge l_n$ is true only in I. Using the chain rule of probability, I can write

$$\pi(I) = \pi(l_1)\pi(l_2|l_1)\pi(l_3|l_1, l_2)\ldots\pi(l_n|l_1, \ldots, l_{n-1}) \tag{6}$$

given any order of the l_i. Let us indicate with $\pi(l|L_{Il})$ a generic factor of $\pi(I)$. Let r be a ranking of the atoms of $H_B(Q)$ so that: (1) Q is acyclic with respect to r and (2) if two clauses of Q have not mutually exclusive bodies they have different ranking for the atoms in the head. Given that the language bias allows only acyclic programs and that only clauses with mutually exclusive bodies can share an atom in the head, such a ranking exists. Let $r(l)$ be the rank of the atom of the literal l. Consider an order of the l_is such that if $r(l) < r(m)$ then l precedes m and if $r(l) = r(m)$ and l is positive and m negative then l precedes m. Such ordering is different for every interpretation, what remains constant among interpretations is the division of literals in layers, i.e., in groups of literals with the same ranking. Inside each layer, the ordering of the literals is different for every interpretation. Note that, for every interpretation and every layer, at most one literal is positive: in fact, if we call l such a literal, there is a single rule R with l in the head that has the body true in I for the assumption of mutual exclusion and all the other atoms in the head of l are false.

Let $W(I)$ be the left member of the interpretation constraint for I and let $A(I)$ the set of atoms a for which the factor p_a is in $W(I)$. We know that, $\forall(I, \pi(I)) \in E$, $A(I) \subseteq I$ and that $W(I) = \pi(I)$.

Consider an atom $a \in A(I)$. Then $p_a < 1$ and there exist a clause R in Q_σ with the body B_a true in I that has atom a in the head. By the way in which p_a was computed then $p_a = \pi(a|B_a)$.

I now prove that $\forall I, a \in A(I)\ \pi(a|B_a) \geq \pi(a|L_{Ia})$. If $\exists I, a\ \pi(a|B_a) < \pi(a|L_{Ia})$, consider an I and an a such that $\pi(a|B_a) < \pi(a|L_{Ia})$ in I and $r(a)$ is the lowest. Then there must exist an atom b such that $\pi(b|B_a) > \pi(b|L_{Ia})$ (b is in the head of the clause with body B_a together with a). Since $\pi(b|B_a) > 0$, such an atom exists. We now prove that there must exist an interpretation J in E where L_{Ia}, B_a and b are true and where $\pi(b|B_a) > \pi(b|L_{Jb})$.

In fact, if such a J would not exist in E, then $\pi(J) = 0$ while $W(J) \neq 0$, against the hypothesis that $W(I) = \pi(I)$, that the $\pi(I)$ sum up to one and that $\sum_{I \in \mathcal{I}_Q, \exists \sigma WFM(Q_\sigma) = I} W(I) = 1$. This can be shown by observing that, since Q is acyclic, the model of an instance of it can be built layer by layer: rule $a \vee b \vee \ldots \leftarrow B_a$ is applicable in L_{Ia} and can produce the interpretation $L_{Ia} \cup \{b\}$. Any model obtained by applying the remaining rules will be a superset of $L_{Ia} \cup \{b\}$.

Thus there exist an interpretation J such that $L_{Ia} = L_{Jb}$, $b \in J$ and $\pi(b|B_a) > \pi(b|L_{Jb})$. If there is no c such that $\pi(c|B_c) < \pi(c|L_{Jc})$ then $W(J)$ would be greater then $\pi(J)$. If there is a c with $r(c) < r(b)$ such that $\pi(c|B_c) < \pi(c|L_{Jc})$, then $\pi(c|B_c) < \pi(c|L_{Ic})$ since $L_{Jc} \subseteq L_{Jb} = L_{Ia} \subseteq I$, against the hypothesis that a is the atom with the lowest rank such that $\pi(a|B_a) < \pi(a|L_{Ia})$.

If there is a c with $r(c) > r(b)$ such that $\pi(c|B_c) < \pi(c|L_{Jc})$ then consider the one with the lowest rank $r(c)$.

Then, in interpretation J, c is the atom with the smallest rank such that $\pi(c|B_c) < \pi(c|L_{Jc})$. As we did for atom b, we can prove that there exist an atom d and an interpretation K such that $L_{Jc} = L_{Kd}$, $d \in K$ and $\pi(d|B_c) > \pi(d|L_{Kd})$ and the reasoning can be applied again. Since the number of atoms and interpretations is finite, this prove the statement.

Now suppose that there exist an atom $a \in I$ such that $\pi(a|L_{Ia}) < 1$ and for which there is no clause of Q with a in the head and the body true in I, i.e. $a \notin A(I)$. In this case $W(I) > \pi(I)$ because no factor would be present in $W(I)$ for a, $\pi(a|B_a) \geq \pi(a|L_{Ia})$ for all the atoms $a \in A(I)$ and all the other factors of $\pi(I)$ are smaller or equal to 1.

If $\pi(a|L_{Ia}) = 1$ then there exist a definite clause with a in the head and the body true in L_{Ia} because the space of definite clause is searched completely.

I will now prove that I is a Fitting model of Q_σ. $T_{Q_\sigma}(I) = I^+$ because all the positive literals a of I have a clause in Q_σ that has a in the head and that has the body true in I and every clause $a \leftarrow B$ with B true in I has a in I. It remains to be proved that $N_{Q_\sigma}(I) = \neg I^-$.

For every $l = \neg a$ in I^- there is no clause with the body true in I that has a in the head, i.e. $\forall a \leftarrow B\ \in Q_\sigma$, $\neg B \cap I \neq \emptyset$. Thus $a \in N_{Q_\sigma}(I)$ and $I^- \subseteq \neg N_{N_\sigma}(I)$.

We now have to prove that $I^- \supseteq \neg N_{Q_\sigma}(I)$. Given an $a \in N_{Q_\sigma}(I), \forall a \leftarrow B \ \in Q_\sigma$, $\neg B \cap I \neq \emptyset$, thus B is false in I and $a \notin I^+$. Since I' is total, then $\neg a \in I^-$.

Since Q_σ is acyclic, then $FM(Q_\sigma)$ is two valued and $FM(Q_\sigma) = WFM(Q_\sigma)$ (theorem 3). Since $I = FM(Q_\sigma)$ then $I = WFM(Q_\sigma)$ and $S(I)$ is not empty.

6 Language Biases

In this section I discuss the language biases available in ALLPAD by first presenting some preliminary definitions and then by presenting the biases themselves.

A couple (G, R) of a set G and a relation R over $G \times G$ is a *quasi-ordered set* if R is reflexive and transitive. I will usually denote a relation R of a quasi-ordered set as $\geq$. I will write that $A > B$ if $A \geq B$ but $B \ngeq A$.

The most common generality relation used in ILP is θ-subsumption. Since it is reflexive and transitive, if G is a set of clauses, $(G, \geq_\theta)$ is a quasi-ordered set. Since I consider ground clauses, θ-subsumption is equivalent to $\subseteq$, i.e., $C \geq_\theta D$ iff $C \subseteq D$

Let us now define the concepts of downward and upward refinement operators for quasi-ordered sets. A *downward refinement operator* for a quasi ordered set $(G, \geq)$ is a function ρ such that $\rho(C) \subseteq \{D | C \geq D\}$ for every $C \in G$.

An *upward refinement operator* for a quasi ordered set $(G, \geq)$ is a function δ such that $\delta(C) \subseteq \{D | D \geq C\}$ for every $C \in G$.

A *language bias* is a specification of the set G, of the relation $\geq$ and of a refinement operator. Sometimes I will use the term language bias to indicate only the specification of the set G.

Let us give also some properties regarding refinement operators. Let $(G, \geq)$ be a quasi-ordered set and let ρ a downward refinement operator for $(G, \geq)$ [26]:

- the sets of *one-step refinements*, *n-step refinements*, and *refinements* of some $C \in G$ are respectively:
  $\rho^1(C) = \rho(C)$
  $\rho^n(C) = \{D |$ there is an $E \in \rho^{n-1}$ such that $D \in \rho(E)\}, n \geq 2$
  $\rho^*(C) = \rho^1(C) \cup \rho^2(C) \cup \rho^3(C) \ldots$
- a *ρ-chain* from C to D is a sequence $C = C_0, C_1, \ldots, C_n = D$, such that $C_i \in \rho(C_{i-1})$ for every $1 \leq i \leq n$
- ρ is *locally finite* iff, for every $C \in G$, $\rho(C)$ is finite and computable
- ρ is *complete* iff, for every $C, D \in G$ such that $C > D$, there is an $E \in \rho^*(C)$ such that $D \approx E$ (i.e. D and E are equivalent in the $\geq$-order)
- ρ is *weakly complete* iff $\rho^*(\text{false}) = G$ [3],
- ρ is *proper* iff, for every $C \in G$, $\rho(C) \subseteq \{D | C > D\}$
- ρ is *ideal* iff it is locally finite, complete and proper
- ρ is *optimal* iff $\forall D, C_1, C_2 \in G : D \in \rho^*(C_1) \cap \rho^*(C_2) \rightarrow C_1 \in \rho^*(C_2)$ or $C_2 \in \rho^*(C_1)$ [8], i.e., each clause in G is visited at most once.

Similar definitions can be given for an upward refinement operator δ.

In an optimal refinement operator, there is exactly one ρ-chain from C to D if $C > D$. This means that the refinement graph becomes a tree, with $false$ as root.

ALLPAD can employ different language biases for the heads and for the bodies of clauses. This is achieved by having refinement operators that work only on the head or only on the body of clauses but never on both.

For bodies, I will consider two possible biases, the $\mathcal{D}$LAB language bias and a language bias inspired to the one of Progol [24] and Aleph [33].

For heads, I will consider two possible language biases: $\mathcal{D}$LAB and a language biases inspired to the one of Progol and Aleph.

Let us start with the language biases for the bodies. The $\mathcal{D}$LAB [9] language bias considers a set G that is finite. It is described by a means of a grammar that contains clause templates that allow to specify membership of literals into sets. For example, consider the following $\mathcal{D}$LAB clause template:

$$\{0 \cdots 2 : [human(john), 1 \cdots 2 : [female(john), male(john)]]\}$$

This grammar describes sets of literals that contain between 0 and 2 literals from the set $[human(john), 1 \cdots 2 : [female(john), male(john)]]$. In its turn, $1 \cdots 2 : [female(john), male(john)]$ stands for a set of literals containing between 1 and 2 literals from the set $[female(john), male(john)]$.

The $\mathcal{D}$LAB refinement operator for a clause template is defined on the basis of the observation that a clause in G can be obtained with a sequence of subset selection operations from sets. Thus, a downward refinement can be obtained by enlarging one of these subsets. A refinement operator is defined as the set of all the possible minimal extensions of these subsets.

The $\mathcal{D}$LAB refinement operator can be proper, depending on how the grammar is written. In fact, if I write a grammar where a literal appears more than once, the literal can be selected for inclusion in a clause when it is already present. It is locally finite, since the set of minimal extensions is finite.

Moreover, the $\mathcal{D}$LAB refinement operator can be optimal, i.e., it can generate each clause of G at most once, depending on user choice. If the optimal switch is set to off, then the refinement operator is complete, otherwise it is only weakly complete.

Let us now consider the other language bias for bodies. This bias consists in having declarations similar to those of Progol and Aleph.

The set G is defined by a *head bias* declaration and by a number of *determination declarations*. The head bias declaration is of the form

$head\_bias(key, AllowedAtoms)$.

where key is an atom unique among all the *head_bias* facts and $AllowedAtoms$ is a list of atoms that can appear together in the head of a clause. The determination declarations are of the form

$determination(key, b)$.

meaning that atom b can appear in the clauses where atoms in $AllowedAtoms$ appears in the head. Let us call G_{det} the set of clauses described by these declarations. If more than one head bias declaration is present then different clause templates are defined.

The refinement operator I consider for this language bias is called ρ_D. It is defined in the following way: given a clause C relative to key id, obtain $C' \in$

$\rho_D(C)$ by adding an atom b to the body of C where there exists a declaration of the form *determination(id,b)* unless b is already in C.

For example, consider the bias

$head\_bias(id1, [p, q])$.

$determination(id1, r)$.

$determination(id1, s)$.

$determination(id1, t)$.

and consider the clause $C = p \vee q \leftarrow t$, then $\rho_D(C)$ is $\{(p \vee q \leftarrow t, r), (p \vee q \leftarrow t, s)\}$.

Let us now show that ρ_D is locally finite, proper and complete (i.e. it is ideal) with respect to the quasi-ordered set $(G_{det}, \subseteq)$ (and therefore also with respect to the quasi-ordered set $(G_{det}, \geq_\theta)$). However, ρ_D is not optimal.

The local finiteness of ρ_D is evident from the fact that the number of determination statements is finite.

ρ_D is proper since it does not add a literal if it is already there, therefore $C \subset D$ for every $D \in \rho_D(C)$

ρ_D is complete since, for every D such that $C \subset D$, D can be reached by adding one at a time the atoms from $D \setminus C$.

That ρ_D is not optimal can be seen from the following example: consider the following language bias

$determination(id1, q)$.

$determination(id1, r)$.

and clause C_1

$$p \leftarrow q$$

clause C_2

$$p \leftarrow r$$

and clause E

$$p \leftarrow q, r$$

then $E \in \rho_D(C_1)$ and $E \in \rho_D(C_2)$ but $C_1 \notin \rho_D^*(C_2)$ nor $C_2 \notin \rho_D^*(C_1)$. In other words, if I start refining from the empty clause, clause E will be visited twice, once coming from clause C_1 and once from clause C_2.

ALLPAD employs for the bodies either the language bias $\mathcal{D}$LAB with the non-optimal operator, or $(G_{det}, \subseteq)$ with ρ_D. ALLPAD uses a non-optimal operator because the search is incomplete, so literals discarded early because they did not lead to good refinements can be considered again later for addition.

For heads, ALLAPD can use either $\mathcal{D}$LAB for top-down search or a bias inspired to one of Progol and Aleph for bottom-up search. In case of top-down search, $\mathcal{D}$LAB with an optimal operator is used. In case of a bottom-up search, the set G is the set of all the heads that are subsets of the set *AllowedAtoms* defined in a head bias declaration, i.e., $G = 2^{AllowedAtoms}$. The generality relation is the subset relation $\supseteq$ since the heads will be ground as well. The refinement operator is called δ_S.

δ_S takes as input a state instead of a set of atoms and returns a set of states. A *state* is a couple (H, L) where H is a head (set of atoms) and L is a set of atoms that can be still removed from H. L is such that $L \subseteq H$. Given a state

(H, L) where $L = \{A_1, A_2, \ldots, A_n\}$, $\delta_S((H, L))$ contains n states (H_i, L_i) for $i = 1, \ldots, n$ where H_i is $H \setminus \{A_i\}$ and L_i is $\{A_{i+1}, A_{i+2}, \ldots, A_n\}$.

The bottom-up search in the space of possible heads starts from a state with both the head and the set of atoms equal to *AllowedAtoms*. Let us see an example. Suppose $AllowedAtoms = \{a, b, c, d\}$. Then the search starts from $S_1 = (\{a, b, c, d\}, \{a, b, c, d\})$. $\delta_S(S_1)$ is

$\{(\{b, c, d\}, \{b, c, d\}),\ (\{a, c, d\}, \{c, d\}),$
$(\{a, b, d\}, \{d\}),\ (\{a, b, c\}, \emptyset)\}$

$\delta_S((\{b, c, d\}, \{b, c, d\}))$ is

$\{(\{c, d\}, \{c, d\}),\ (\{b, d\}, \{d\}),\ (\{b, c\}, \emptyset)\}$

δ_S is locally finite, proper, not complete but weakly complete with respect to the quasi-order $(2^{AllowedAtoms}, \supseteq)$. Moreover, it is optimal, i.e., every head in G is visited once.

Lemma 11. *δ_S is optimal.*

Proof. Given two heads H_1 and H_2 such that $H_1 \supset H_2$, I will show that there is only one δ_S-chain from a state $S_1 = (H_1, L_1)$ to $S_2 = (H_2, L_2)$.

In order to be able to go from S_1 to S_2, L_1 must be such that $L_1 \supseteq H_1 \setminus H_2$. So let us start from a S_1 state with such an L_1. Let $H_1 \setminus H_2$ be $\{h_1, h_2, \ldots, h_n\}$.

In order to go from S_1 to S_2 I must perform n refining steps, at each step I must remove one atom from H_1. Let us consider the first step: suppose I consider an ordering of the atoms in L_1 where h_1 is the first atom appearing in L_1 from the set $H_1 \setminus H_2$. Removing h_1 I obtain the state $S_1' = (H_1 \setminus \{h_1\}, L_1 \setminus (J \cup \{h_1\}))$ where J is the set of atoms preceding h_1 in the ordering of L_1 .

All the other states $S = (H, L)$ in $\delta_S(S_1)$ are such that from them H_2 can not be reached. In fact, consider the states generated by removing atoms that appear before h_1 in the ordering chosen for L_1. These states are such that H_2 can not be reached anymore because H is no more a superset of H_2. Consider the states generated by removing atoms that appear after h_1 in ordering chosen for L_1. In this case, H still contains h_1 but L does not contain anymore h_1 so it cannot be removed anymore from H. Since the ordering considered for L_1 is general, this is true for all the possible orderings.

A similar reasoning can be applied for all the n steps therefore there is only one δ_S-chain from S_1 to S_2.

A similar proof can be used to show that δ_S is weakly complete.

Optimal operators are used in the search in the space of heads because the search is complete, therefore every clause is visited at most once.

7 Experiments

ALLPAD was applied to the problem of predicting the tertiary structure of proteins by classifying them into one of the SCOP classes [35]. Each protein is described by a sequence of secondary structure elements. The sequence is represented in first order logic as an interpretation where each atom is either of the

form *he(Type,Length,Position)* or of the form *st(Orientation,Length,Position)*. The last argument is an ordinal number indicating the position in the sequence. The first atom form indicates that the element is an helix and specifies its type and length. The possible types are *h(left,alpha)*, *h(right,alpha)*, *h(left,gamma)*, *h(right,gamma)*, *h(left,omega)*, *h(right,omega)*, *h(right,pi)*, *h(right,f3to10)*, *27ribbon* and *ploy proline*. The second atom form indicates that the element is a strand and specifies its orientation and length. The possible orientations are *null* (the beginning of a strand), *plus* (a parallel strand of a sheet) or *minus* (an anti-parallel strand of a sheet). The length of helices and strands is defined as the number of amino acids they are composed of and was discretized by diving the range into three intervals of equal length.

The dataset available [21] (kindly provided by Kristian Kersting) regards the prediction of domains at the second level of the SCOP hierarchy, namely the level of folds. In particular, the data regards the alpha beta protein class (a/b) and, in this class, the five most populated subclasses (i.e. folds) are considered: TIM beta/alpha-barrel, NAD(P)-binding Rossmann-fold domains, Ribosomal protein L4, glucosamine 5-phosphate deaminase/isomerase and leucine aminopeptidas. The folds will be respectively indicated in the following with the names fold1, fold2, fold23, fold37 and fold55.

The dataset available has 721 examples for class fold1, 360 for fold2, 274 for fold 23, 441 for fold37 and 290 for fold55. In the dataset, only two helix types are actually present, namely *h(right, alpha)* and *h(right,f3to10)*.

ALLAPD can be used for classification as other probabilistic model learners: a model is learned for each class using only the examples for that class and a test case is assigned the class whose model gives the highest probability to the case.

In order to learn an LPAD that describes a class, each interpretation given as input to the system is annotated with the same probability, given by one over the total number of interpretations in the training set, since no interpretation appears more than once.

Proteins are modeled with LPADs as stochastic processes: I want to predict the structure at position p on the basis of the structures in the k previous positions. To this purpose, ALLPAD learns programs containing rules having all the possible structures with position equal to p in the head and a conjunction of structures in the body with positions belonging to the set $S(p,k) = \{p-1, p-2, \ldots, p-k\}$ for a given k. Obviously I can not have two different structures for the same position $t \in S(p,k)$ in the body since in that case the body would be false in every interpretation. An example of such a rule is:

```
helix(h(right,alpha),long,7):0.571 ;
helix(h(right,f3to10),short,7):0.429 :-
  helix(h(right,alpha),long,5),
  strand(null,medium,4).
```

Therefore I give ALLPAD a language bias expressed using DLAB that contains a rule template for each position p from 1 to the maximum length of the protein

sequences in the training set. Each rule template allows each possible structure in the head at position p and each possible structure in the body for positions $p-1$, $p-2$ up to position $p-k$, if they are larger or equal to 1.

Since the optimization phase finds only an approximate solution, I must use an approximate procedure for testing the theories learned. When the test case is a model of an instance of the program, its probability can be computed in the following way: all the disjunctive rules whose bodies are true in the interpretation are identified and, for each such rule, the single head that is true in the interpretation is identified. The probability of the interpretation is obtained by multiplying the probabilities associated to all the head atoms identified in this way. For each position, there is a single rule whose body is true in the interpretation because the rules have mutually exclusive bodies.

When the test case is not a model of an instance of the program, the probability assigned to the case is 0. This happens for example when for one or more positions there is no rule with the body true in the interpretation. Since rules may be missing because the solution is approximate, in the approximate testing procedure I adopt the following approach: if for a position no rule is applicable, the probability for the structure in that position is given by the conditional probability of the structure given the position and the class. This corresponds to using a rule with an empty body (i.e. a default rule).

The accuracy of the learned LPADs is compared with the accuracy of a naive Bayes classifier obtained by applying the approximate testing procedure so that the conditional probability given the position and the class is used for all positions.

Two experiments were performed using 10-fold cross validation. In both experiments I used Xpress-Optimizer by Dash Optimization for solving the MIP problem. This tool allows the user to set a time limit to the optimization. The time limit has been set to 1 hour for each class in the first experiment and to 100 minutes for each class in the second experiment.

The other important parameters are: the value of k (the number of previous positions to consider), set to 4; the size of the beam, set to 100, and the maximum number of bodies to be refined for each clause template, set to 100 in the first experiment and to 125 in the second experiment for all classes and cross-validation folds apart from two folds for class fold1 and one fold for class fold37 where it was set to 115 because in 100 minutes no admissible solution was found for the MIP problem.

All the experiments have been performed on a PC with an Athlon XP 2600+ processor at 2138 Mhz, 1GB of RAM and Windows 2000.

The obtained results are summarized in Table 1. LLPAD has been tested as well on the dataset but the constraint satisfaction problem lead to an insufficient memory error in all cases.

A cross-validated paired two-tailed t test has been performed for comparing the accuracy of ALLPAD with that of naive Bayes. The null hypothesis of equivalence can be rejected with the probability indicated in the Significance column in Table 1. The last column of the table shows the average number of default

Table 1. Results of the experiments.

Experiment	Av. acc.	St. dev.	Significance	Av. def. rules
Naive Bayes	82.79%	0.03087	-	20.09
First	85.14%	0.01990	98.3%	16.83
Second	85.67%	0.02394	98.6%	16.64

rules used in testing. In the case of naive Bayes, this is the average length of the sequences. This means that, on average, 3.26 learned rules were used in the testing phase of the first experiment and 3.45 learned rules were used in the testing phase of the second experiment. The accuracy improvement between the first and second experiment, even if it is significant only with probability 65.9%, shows that the results can be improved by employing more computation time.

As regards the execution times, they were dominated by the optimization problem: the other phases only took few minutes per fold.

8 Related Works

There has been recently a growing interest in the field of probabilistic logic programming: a number of works have appeared that combine logic programming or relational representations with probabilistic reasoning. Among these works, I cite: Bayesian Logic Programs (BLPs) [17, 18], Probabilistic Relational Models (PRMs) [14], Independent Choice Logic (ICL) [27], Stochastic Logic Programs [25, 7], the Meta Interpreter Approach (MIA) [5], Logical Bayesian Networks (LBNs) [10] and CLP(BN) [31].

In [4] the author compares MIA, CLP(BN), BLPs, LBNs and LPADs on the problem of representing Mendel's law of inheritance of pea color. The author concludes that LPADs are the only ones that are both among the most readable and among those that are able to express the most stringent constraints on the joint distribution by using only qualitative information.

In [37] the authors compare LPAD with Bayesian Logic Programs [17, 18]. They show that every BLP can be expressed as an LPAD with a semantics that matches that of the BLP. Moreover, they also show that a large subset of LPADs can be translated into BLPs in a way that preserves the semantics of the LPADs. Such a subset is the set of all finite ground LPADs that are acyclic. Therefore, the techniques developed in [19] for learning BLPs can be used for learning this class of LPADs as well and, on the other side, ALLPAD can be used for learning BLPs, thus representing an alternative approach to those in [19].

Probabilistic Relational Models [14] extend the formalism of Bayesian networks in order to model domains that are described by a multi-table relational database. Each attribute of a table is considered as a random variable and its set of parents can contain other attributes of the same table or attributes of other tables connected to the attribute table by foreign key connections. The relationship between PRM and LPADs is not clear. For sure they have a non-empty

intersection: the PRM that do not contain attributes that depend on aggregate functions of attributes of other tables can be expressed as LPADs. Moreover, LPADs are not a subset of PRM since they can express partial knowledge regarding the dependence of an attribute from other attributes, in the sense that with LPADs it is possible to specify only a part of a conditional probability table. Therefore the learning techniques developed in [14] can not be used in substitution of the techniques proposed in this paper.

Stochastic Logic Programs (SLPs) [7, 25] are another formalism integrating logic and probability. In [37] the authors have shown that a SLP can be translated in LPAD, while whether the opposite is possible is not known yet. Therefore, it is not clear at the moment whether the techniques used for learning SLPs can be used for learning LPADs.

PRISM [32] is a logic programming language in which a program is composed of a set of facts and a set of rules such that no atom in the set of facts appears in the head of a rule. In a PRISM program, each atom is seen as a random variable taking value true or false. A probability distribution for the atoms appearing in the head of rules is inferred from a given probability distribution for the set of facts. PRISM differs from LPADs because PRISM assigns a probability distribution to ground facts while LPADs assign a probability distribution to the literals in the head of rules. PRISM programs resemble programs of ICL, in the sense that PRISM facts can be seen as ICL abducibles. In [32] the author also proposes an algorithm for learning the parameters of the probability distribution of facts from a given probability distribution for the observable atoms (atoms in the head of rules). However, no algorithm for learning the rules of a PRISM program has been defined. Inferring the parameters of the distribution is performed in LLPAD analytically by means of theorem 5 rather than by means of the EM algorithm as in PRISM.

Considering the problem of learning LPADs, a system related to ALLPAD is *Classic'cl* [34]. ALLPAD differs from *Classic'cl* in following respects: it is able to solve the constraint problem in an approximate way, it learns definite clauses separately, it can search the space of heads also bottom-up and it adopts beam search in the space of bodies.

As regards the classification of proteins in SCOP classes, the first work applying ILP to the problem was [35] where the authors obtain an average accuracy of 78.28% using Progol. However, their the dataseet is very different because they consider other classes besides the alpha beta class. [21, 16, 20, 15] use a dataset similar to the one used here. [21] and [20] use logical hidden Markov models and achieve respectively accuracies of 74% and 76%. [16] uses Fisher kernels for logical sequences and achieves an accuracy of 83.6%. Finally [15] uses conditional random fields for logical sequences and achieves an accuracy of 92.96%. The results of ALLPAD compare favourably with all the previous results apart from the last one, even if the system is not specifically tailored to learning sequences, as all previous systems are apart from [35].

9 Conclusion and Future Works

The learning algorithm ALLPAD has been presented. It improves LLPAD by solving the constraint satisfaction problem in an approximate way so that real world problems can be solved.

ALLPAD has been tested on the problem of classifying proteins into SCOP classes and has shown an accuracy significantly superior to a naive Bayes approach and superior to all previous approaches to the same problem apart from [15].

In the future, work will be devoted to the definition of a generality relation among LPAD clauses and of the relative generalization operators, so that the ground clauses that are returned by ALLPAD can be generalized in a sixth phase.

10 Acknowledgements

This work was partially funded by the Ministero dell'Istruzione, della Ricerca e dell'Università under the PRIN 2005 project "Specification and verification of agent interaction protocols". The author would like to thank Evelina Lamma for having read a draft of the paper and Marco Gavanelli for his helpful suggestions regarding MIP.

References

1. K. R. Apt and M. Bezem. Acyclic programs. *New Generation Comput.*, 9(3/4):335–364, 1991.
2. Krzysztof R. Apt and Roland N. Bol. Logic programming and negation: A survey. *J. Log. Program.*, 19/20:9–71, 1994.
3. L. Badea and M. Stanciu. Refinement operators can be (weakly) perfect. In S. Džeroski and P. Flach, editors, *Proceedings of the 9th International Workshop on Inductive Logic Programming*, volume 1634 of *Lecture Notes in Artificial Intelligence*, pages 21–32. Springer-Verlag, 1999.
4. H. Blockeel. Probabilistic logical models for mendel's experiments: An exercise. In *Proceedings of the Fourteenth International Conference on Inductive Logic Programming, Work in Progress Track*, 2004.
5. Hendrik Blockeel. Prolog for first-order bayesian networks: A meta-intepreter approach. In *Multi-Relational Data Mining (MRDM03)*, 2003.
6. K. L. Clark. Negation as failure. In *Logic and Databases*. Plenum Press, 1978.
7. James Cussens. Stochastic logic programs: Sampling, inference and applications. In *Sixteenth Annual Conference on Uncertainty in Artificial Intelligence (UAI-2000)*, pages 115–122, San Francisco, CA, 2000. Morgan Kaufmann.
8. L. De Raedt and L. Dehaspe. Clausal discovery. *Machine Learning*, 26(2–3):99–146, 1997.
9. L. Dehaspe and L. De Raedt. DLAB: A declarative language bias formalism. In *Proceedings of the 10th International Symposium on Methodologies for Intelligent Systems*, volume 1079 of *Lecture Notes in Artificial Intelligence*, pages 613–622. Springer-Verlag, 1996.

10. D. Fierens, H. Blockeel, J. Ramon, and M. Bruynooghe. Logical bayesian networks. In *Third Workshop on Multi-Relational Data Mining*, pages 19–30, 2004.
11. M. Fitting. A kripke-kleene semantics for logic programs. *Journal of Logic Programming*, 2(4):295–312, 1985.
12. M. Fitting. A kripke-kleene semantics for logic programs. *J. Log. Program.*, 2(4):295–312, 1985.
13. M. Gelfond and V. Lifschitz. The stable model semantics for logic programming. In R. Kowalski and K. A. Bowen, editors, *Proceedings of the 5th Int. Conf. on Logic Programming*, pages 1070–1080. MIT Press, 1988.
14. L. Getoor, N. Friedman, D. Koller, and A. Pfeffer. Learning probabilistic relational models. In Saso Dzeroski and Nada Lavrac, editors, *Relational Data Mining*. Springer-Verlag, Berlin, 2001.
15. B. Gutmann and K. Kersting. TildeCRF: Conditional random fields for logical sequences. In *Seventeenth European Conference on Machine Learning*, Berlin, Germany, 2006. Springer.
16. K. Kersting and T. Gärtner. Fisher kernels for logical sequences. In *Fifteenth European Conference on Machine Learning*, pages 205–216, Berlin, Germany, 2004. Springer.
17. K. Kersting and L. De Raedt. Bayesian logic programs. In *Work-in-Progress Reports of the Tenth International Conference on Inductive Logic Programming (ILP2000), London, UK*, 2000.
18. K. Kersting and L. De Raedt. Bayesian logic programs. Technical Report 151, Institute for Computer Science, University of Freiburg, Freiburg, Germany, April 2001.
19. K. Kersting and L. De Raedt. Towards combining inductive logic programming and bayesian networks. In C. Rouveirol and M. Sebag, editors, *Eleventh International Conference on Inductive Logic Programming (ILP-2001), Strasbourg, France, September 2001*, number 2157 in LNAI. Springer-Verlag, 2001.
20. K. Kersting, L. De Raedt, and T. Raiko. Logical hidden markov models. *Journal of Artificial Intelligence Research*, 25:425–456, 2006.
21. K. Kersting, T. Raiko, S. Kramer, and L. De Raedt. Towards discovering structural signatures of protein folds based on logical hidden markov models. In *Pacific Symposium on Biocomputing*, pages 192–203, Singapore, 2003. World Scientific Press.
22. J. W. Lloyd. *Foundations of Logic Programming*. Springer-Verlag, 2nd extended edition, 1987.
23. J. Lobo, J. Minker, and A. Rajasekar. *Foundations of Disjunctive Logic Programming*. MIT Press, Cambridge, Massachusetts, 1992.
24. S. Muggleton. Inverting entailment and Progol. In *Machine Intelligence*, volume 14, pages 133–188. Oxford University Press, 1995.
25. S. H. Muggleton. Learning stochastic logic programs. *Electronic Transactions in Artificial Intelligence*, 4(041), 2000.
26. Shan-Hwei Nienhuys-Cheng and Ronald de Wolf. *Foundations of Inductive Logic Programming*, volume 1228 of *Lecture Notes in Artificial Intelligence*. Springer, Berlin, Germany, 1997.
27. D. Poole. The Independent Choice Logic for modelling multiple agents under uncertainty. *Artificial Intelligence*, 94(1–2):7–56, 1997.
28. T. C. Przymusinski. On the declarative semantics of deductive databases and logic programs. In J. Minker, editor, *Foundations of deductive databases and logic programming*. Morgan Kaufmann Publishers Inc., San Francisco, CA, 1988.

29. F. Riguzzi. Learning logic programs with annotated disjunctions. In *Fourteenth International Conference on Inductive Logic Programming*, pages 270–287, Berlin, Germany, 2004. Springer.
30. Fabrizio Riguzzi. Learning logic programs with annotated disjunctions. In A Srinivasan and R. King, editors, *Inductive Logic Programming: 14th International Conference, ILP 2004, Porto, Portugal, September 6-8, 2004. Proceedings*, number 3194 in Lecture Notes in Artificial Intelligence, pages 270–287, Heidelberg, Germany, September 2004. Springer Verlag.
31. V. Santos Costa, D. Page, M. Qazi, and J. Cussens. Clp($\mathcal{BN}$): Constraint logic programming for probabilistic knowledge. In *Uncertainty in Artificial Intelligence (UAI03)*, 2003.
32. T. Sato. A statistical learning method for logic programs with distribution semantics. In *12th International Conference on Logic Programming (ICLP95)*, pages 715–729, 1995.
33. A. Srinivasan. Aleph, 2004. URL: http://web.comlab.ox.ac.uk/oucl/research/ areas/machlearn/Aleph/aleph_toc.html.
34. C. Stolle, A. Karwath, and L. De Raedt. *Cassic'cl*: An integrated ILP system. In *The Eigth International Conference on Discovery Science*, Berlin, Germany, 2005. Springer.
35. M. Turcotte, S. Muggleton, and M. J. E. Sternberg. The effect of relational background knowledge on learning of protein three-dimensional fold signatures. *Machine Learning*, 43(1/2):81–95, 2001.
36. A. Van Gelder, K. A. Ross, and J. S. Schlipf. The well-founded semantics for general logic programs. *Journal of the ACM*, 38(3):620–650, 1991.
37. J. Vennekens and S. Verbaeten. Logic programs with annotated disjunctions. Technical Report CW386, K. U. Leuven, 2003. http://www.cs.kuleuven.ac.be/ ∼joost/techrep.ps.
38. J. Vennekens, S. Verbaeten, and M. Bruynooghe. Logic programs with annotated disjunctions. In *The 20th International Conference on Logic Programming (ICLP04)*, 2004.