# Induction of Recursive Theories in the Normal ILP Setting: Issues and Solutions

Floriana Esposito, Donato Malerba, and Francesca A. Lisi

Dipartimento di Informatica, Università degli Studi di Bari,
Via Orabona 4, I-70126 Bari, Italy
{esposito | malerba | lisi}@di.uniba.it

**Abstract.** Induction of recursive theories in the normal ILP setting is a complex task because of the non-monotonicity of the consistency property. In this paper we propose computational solutions to some relevant issues raised by the multiple predicate learning problem. A separate-and-parallel-conquer search strategy is adopted to interleave the learning of clauses supplying predicates with mutually recursive definitions. A novel generality order to be imposed to the search space of clauses is investigated in order to cope with recursion in a more suitable way. The consistency recovery is performed by reformulating the current theory and by applying a layering technique based on the collapsed dependency graph. The proposed approach has been implemented in the ILP system ATRE and tested in the specific context of the document understanding problem within the WISDOM project. Experimental results are discussed and future directions are drawn.

## 1 Introduction

Inductive learning of recursive logical theories is equivalent to learning multiple predicate definitions from a set of examples. De Raedt *et al.* [9] have showed that learning multiple predicates is more difficult than learning a single predicate. In fact, the former task is not limited to the generation of several *independent* predicate definitions, but involves the discovery of concept dependencies. A wrong hypothesis on concept dependencies may significantly affect the learning results. Moreover, the ordering typically used in inductive logic programming (ILP), namely $\theta$-subsumption [26], is not sufficient to guarantee the completeness and consistency of learned definitions with respect to logical entailment.

The main problems raised by multiple/recursive predicate learning can be explained in terms of an important property of the normal ILP problem setting: Whenever two individual clauses are consistent on the data, their conjunction need not to be consistent on the same data [11]. As a consequence, clauses supplying predicates with multiple/recursive definitions should not be learned individually but, in principle, they should be generated all together.

In order to overcome these problems, it has been proposed to work on a *weak* setting of ILP [14], in which the monotonicity property is satisfied: Whenever two individual clauses are valid on the data, their conjunction will also be valid on the

J. Cussens and A. Frisch (Eds.): ILP 2000, LNAI 1866, pp. 93-111, 2000.

data. In this setting, clauses can be investigated independently of each other since their interactions are no longer important [10]. On the other hand, the weak setting produces properties of examples instead of rules generating examples. This kind of hypotheses cannot always be used for predicting the truth values of facts. When we are interested in predictions, then the normal ILP setting is more appropriate.

Several studies on the problem of learning restricted forms of recursive theories in the normal ILP setting have been presented in the literature. Cohen [7] proves positive and negative results on the pac-learnability of several classes of logic theories that are allowed to include a recursive clause. Cameron-Jones and Quinlan [5] investigate a heuristic method for preventing infinite recursion in single predicate definitions with the system FOIL. De Raedt *et al.* [9] propose an algorithm, named MPL, that performs a greedy hill-climbing search for learning multiple predicate definitions. Giordana *et al.* [16] define a bottom-up learning algorithm, called RTL, that first learns a hierarchical (i.e., non-recursive) theory $T$ which is complete and consistent, and then tries to synthesize a simple recursive theory from $T$. Aha *et al.* [1] have developed a system called CRUSTACEAN which is able to learn recursive definitions consisting of one unit clause and a two-literals recursive clause. Boström [3] proposes an algorithm that, under some assumptions, correctly specializes a given recursive theory with respect to positive and negative examples. Idestam-Almquist [17] suggests a technique to efficiently learn recursive definitions including one base clause and one tail recursive clause from a random sample of examples. An iterative bootstrap induction method for learning recursive predicate definitions has been studied by Jorge and Brazdil [18]. Finally, Mofizur and Numao [23] adopt a top-down approach to learning recursive programs with only one recursive clause. A thorough overview of achievements in the inductive synthesis of recursive logic programs can be found in [15].

In this paper, a new approach to the problem of learning multiple dependent concepts is proposed. It differs from De Raedt *et al.*'s approach in three aspects: the learning strategy, the generalization model, and the strategy to recover the consistency property of the learned theory when a new clause is added. These ideas have been implemented in the learning system ATRE [21] and tested in the specific context of the document understanding problem within the WISDOM project. In fact, the rules induced by ATRE are used by the document analysis and recognition system WISDOM++ [13] in order to recognize semantically relevant layout components (also called *logical components*) in documents being processed.

The paper is organized as follows. Section 2 introduces the issues related to the induction of recursive logical theories. Section 3 illustrates the learning strategy adopted by ATRE. Section 4 is devoted to the generalization model whose implementation is also sketched. A solution to the problem of recovering non-monotonic theories is proposed in Section 5. In Section 6 the application of ATRE to the document understanding problem and experimental results obtained on a set of real-world multi-page documents are described. Finally, in Section 7 our conclusions are drawn.

## 2 Recursive Theories: Learning Issues

Henceforth, the term *logical theory* will denote a set of definite clauses. Every logical theory $T$ can be associated with a directed graph $\gamma(T)=<N,E>$, called the *dependency graph* of $T$, in which (i) each predicate of $T$ is a node in $N$ and (ii) there is an arc in $E$ directed from a node $a$ to a node $b$ iff there exists a clause $C$ in $T$ such that $a$ and $b$ are the predicates of a positive literal occurring in the head and in the body of $C$, respectively.

A dependency graph allows to represent the predicate dependencies of $T$, where a *predicate dependency* is defined as follows:

**Definition 1 (predicate dependency** [8]**)**. A predicate $p$ depends on a predicate $q$ in a theory $T$ iff (i) there exists a clause $C$ for $p$ in $T$ such that $q$ occurs in the body of $C$; or (ii) there exists a clause $C$ for $p$ in $T$ with some predicate $r$ in the body of $C$ that depends on $q$.

It is straightforward to notice that the direct (i) and indirect (ii) predicate dependencies of $T$ are represented as arcs and paths respectively in $\gamma(T)$. The correspondence may be highlighted by reformulating the problem from an algebraic point of view. Let $\pi(T)$ be the set of predicates occurring in the logical theory $T$. The direct (i) predicate dependencies in $T$ may be mathematically depicted as instances of a binary relation on $\pi(T)$, namely $R_{dpd} \subseteq \pi(T)*\pi(T)$. The binary relation $R_{pd}$ for predicate dependencies is the transitive closure of $R_{dpd}$. Given that each binary relation can be associated with a directed graph, the graph corresponding to $R_{dpd}$ is just the dependency graph $\gamma(T)=<\pi(T),R_{dpd}>$.

**Definition 2 (recursive theory).** A logical theory $T$ is *recursive* if the dependency graph $\gamma(T)$ contains at least one cycle.

In *simple* recursive theories all cycles in the dependency graph go from a predicate $p$ into $p$ itself, that is simple recursive theories may contain recursive clauses, but cannot express mutual recursion. An example of dependency graph for a recursive theory is given in Fig. 1.

odd(succ(X)) ← even(X)
even(succ(X)) ← odd(X)
even(X) ← zero(X)

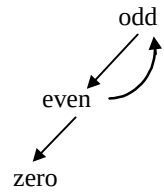



**Fig. 1.** A recursive theory and its corresponding dependency graph for the predicates *even* and *odd*.

Most studies on the problem of induction of recursive theories have been concentrated on learning a simple recursive predicate definition [5, 17, 18, 23]. In this case, the main issue is how to guarantee that learned definitions are intensionally complete and consistent. Learning simple recursive theories is more complicated, since it is necessary to discover the right order in which predicates should be learned [16], that is the dependency graph of the theory. Once such order has been

determined, possibly using statistical techniques, the problem can be boiled down to learning single predicate definitions [20]. The learning problem becomes harder for recursive theories, because the learning of one (possibly recursive) predicate definition should be *interleaved* with the learning of the other ones. One way to build such interleaving is by parallel learning clauses for different predicates. This is, indeed, the strategy adopted by ATRE.

## 3 The Learning Strategy

The high-level learning algorithm in ATRE belongs to the family of *sequential covering* (or *separate-and-conquer*) algorithms [22] since it is based on the strategy of learning one clause at a time (procedure LEARN-ONE-RULE), removing the covered examples and iterating the process on the remaining examples. Indeed, a recursive theory $T$ is built step by step, starting from an empty theory $T_0$, and adding a new clause at each step. In this way we get a sequence of theories

$$T_0 = \varnothing, T_1, \ldots, T_i, T_{i+1}, \ldots, T_n = T$$

such that $T_{i+1} = T_i \cup \{C\}$ for some clause $C$, and all theories in the sequence are consistent with respect to the training set. If we denote with $LHM(T_i)$ the least Herbrand model of a theory $T_i$, the stepwise construction of theories entails that $LHM(T_i) \subseteq LHM(T_{i+1})$, for each $i \in \{0, 1, \ldots, n\text{-}1\}$. Indeed the addition of a clause to a theory can only augment the least Herbrand model of the theory. Henceforth we will assume that both positive and negative examples of predicates to be learned are represented as ground atoms with a + or - label. Therefore examples may or may not be elements of the Herbrand models $LHM(T_i)$.

Let $pos(LHM(T_i))$ and $neg(LHM(T_i))$ be the number of positive and negative examples in $LHM(T_i)$, respectively. If we guarantee that $pos(LHM(T_i)) < pos(LHM(T_{i+1}))$ for each $i \in \{0, 1, \ldots, n\text{-}1\}$, and that $neg(LHM(T_i))=0$ for each $i \in \{0, 1, \ldots, n\}$, then after a finite number of steps a theory $T$, which is complete and consistent, is built. Whether the theory $T$ is "correct", that is whether it classifies correctly all other examples *not* in the training set, cannot be established, since no information on the generalization accuracy can be drawn from the same training data. In fact, the selection of the "best" theory is always made on the basis of an inductive bias embedded in some heuristic function or explicitly expressed by the user of the learning system (preference criterion).

In order to guarantee the first condition above, namely $pos(LHM(T_i)) < pos(LHM(T_{i+1}))$, we suggest to proceed as follows. First, a positive example $e^+$ of a predicate $p$ to be learned is selected, such that $e^+$ is not in $LHM(T_i)$. The example $e^+$ is called *seed*. Then the space of definite clauses more general than $e^+$ is explored, looking for a clause $C$, if any, such that $neg(LHM(T_i \cup \{C\}))=0$. In this way we guarantee that the second condition above holds as well. When found, $C$ is added to $T_i$ giving $T_{i+1}$. If some positive examples are not included in $LHM(T_{i+1})$ then a new seed is selected and the process is repeated.

The most relevant novelties of the learning strategy sketched above are embedded in the design of the procedure LEARN-ONE-RULE being proposed. Indeed, it implements a parallel general-to-specific example-driven search strategy in the space of definite clauses whose ordering (called *generalized implication*) is explained in

Section 4. The search space is actually a forest of as many search-trees (called *specialization hierarchies*) as the number of chosen seeds, where at least one seed per incomplete predicate definition is kept. Each search-tree is rooted with a unit clause and ordered by generalized implication. The forest can be processed in parallel by as many concurrent tasks as the number of search-trees. Each task traverses the specialization hierarchies top-down (or general-to-specific), but synchronizes traversal with the other tasks at each level. Initially, some clauses at depth one in the forest are examined concurrently. Each task is actually free to adopt its own search strategy, and to decide which clauses are worth to be tested. If none of the tested clauses is consistent, clauses at depth two are considered. Search proceeds towards deeper and deeper levels of the specialization hierarchies until at least one consistent clause is found. Task synchronization is performed after that all "relevant" clauses at the same depth have been examined. A supervisor task decides whether the search should carry on or not on the basis of the results returned by the concurrent tasks. When the search is stopped, the supervisor selects the "best" consistent clause according to the user's preference criterion. This strategy has the advantage that simpler consistent clauses are found first, independently of the predicates to be learned. Moreover, the synchronization allows tasks to save much computational effort when the distribution of consistent clauses in the levels of the different search-trees is uneven. The parallel exploration of the specialization hierarchies for the concepts *even* and *odd* is shown in Fig. 2.

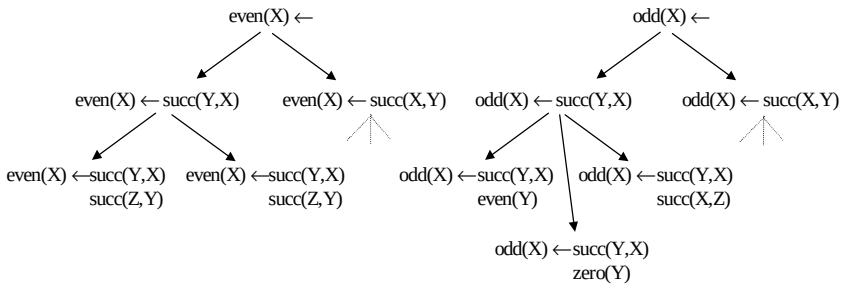



**Fig. 2.** Parallel search for the concepts *even* and *odd*.

To sum up, this *separate-and-parallel-conquer* search strategy provides us with a solution to the problem of interleaving the induction process for distinct predicate definitions. A different approach has been followed in MPL [9], which performs a greedy hill-climbing search for theories and a beam search for each single clause. Clauses can be generated by means of two types of refinement operators, one for the body and one for the head. In particular, it is possible to generate the body of a clause without specifying its head. The generation of clauses like $p \leftarrow p$ is not forbidden, and it is possible to learn a recursive clause for a predicate $p$, before that a base clause for $p$ has been found. In fact, the algorithm uses a depth-bounded interpreter for checking whether an induced theory logically entails an example.

## 4 The Generalization Model

A more precise definition of the search space of the LEARN-ONE-RULE stage is necessary. A generality order (or *generalization model*) provides a basis for organizing this search space. It can be proven that Buntine's *generalized subsumption* [4] is an order suitable for *simple* recursive theories, since it is neither too strong nor too weak.[1]

**Definition 3 (generalized subsumption).** Let $C$ and $D$ be two definite clauses with disjoint variables:

$C$: $C_0 \leftarrow C_1, C_2, \ldots, C_n$

$D$: $D_0 \leftarrow D_1, D_2, \ldots, D_m$

Then $C$ is *more general than D* under generalized subsumption with respect to a theory $T$, denoted $C \leq_T D$, if there exists a substitution $\sigma$ such that $C_0\sigma = D_0$ and for each substitution $\theta$ that grounds the variables in $D$ using new constants not occurring in $C$, $D$, and $T$, it happens that:

$$T \cup \{D_1\theta \leftarrow, D_2\theta \leftarrow, \ldots, D_m\theta\} \vdash_{SLD} (\leftarrow C_1, C_2, \ldots, C_n)\sigma\theta.$$

Unfortunately generalized subsumption is too weak for recursive theories. Thus, we may resort to Plotkin's notion of *relative generalization* [26, 27].

**Definition 4 (relative generalization).** Let $C$ and $D$ be two definite clauses. Then $C$ is *more general than D* under relative generalization with respect to a theory $T$ if there exists a substitution $\theta$ such that $T \models \forall (C\theta \Rightarrow D)$.

However, relative generalization is too strong for our goals. Taken literally, it would lead us to consider unintuitive solutions of the conquer stage as illustrated in the following example.

**Example** Let us consider the following examples:

+: $\{p(a), p(b)\}$

—: $\{p(c)\}$

the following background knowledge:

*BK*: $\{q(a), r(a,b), s(b), r(c,b)\}$

and the following incomplete theory built at the first step:

$T_1$: $p(X) \leftarrow q(X)$

Let $p(b)$ be the selected seed. A desirable search space of clauses more general than $p(b)$ given $BK \cup T_1$ would be the set of clauses whose head is $p(X)$, since our aim is to induce a predicate definition for $p$. This space contains, for instance, the clause $p(X) \leftarrow s(X)$. However, the space of clauses that are relatively more general than $p(b)$ given $BK \cup T_1$ includes other solutions, such as $q(Y) \leftarrow s(Y)$. This last clause is correct and even relatively more general than $p(X) \leftarrow s(X)$. Nevertheless it is unacceptable from an intuitive point of view because it seems not to be related to the target predicate $p$. In this work, we do not consider these less intuitive solutions.

This restriction is reflected by an ordering, named *generalized implication*, which is a special case of relative generalization.

---

[1] Informally, an order is too strong for a class $L$ of theories when it can be used to organize theories of a strictly wider class $L' \supset L$ according to logical entailment. If the organization of theories in $L$ is not consistent with logical entailment, then the order is too weak.

**Definition 5 (generalized implication).** Let $C$ and $D$ be two definite clauses. Then $C$ is *more general than* $D$ under generalized implication with respect to a theory $T$, denoted as $C \leq_{T,\Rightarrow} D$, if there exists a substitution $\theta$ such that $head(C)\theta = head(D)$ and $T \models \forall (C\theta \Rightarrow D)$.

This generality order can be proved to be strictly weaker than relative generalization and strictly stronger than generalized subsumption. Moreover, the following properties hold.

**Proposition 1.** *Let C, D and E be definite clauses and T a theory. The generalized implication order satisfy the properties of:*

*i) reflexivity:* $C \leq_{T,\Rightarrow} C$

*ii) transitivity:* $C \leq_{T,\Rightarrow} D$ *and* $D \leq_{T,\Rightarrow} E$ *then* $C \leq_{T,\Rightarrow} E$

***Proof*** i) Trivial if the empty substitution is chosen. ii) By definition, there exists a substitution $\theta_1$ such that $head(C)\theta_1 = head(D)$ and $T \models \forall(C\theta_1 \Rightarrow D)$ and there exists a substitution $\theta_2$ such that $head(D)\theta_2 = head(E)$ and $T \models \forall(D\theta_2 \Rightarrow E)$. Let $\theta = \theta_1\theta_2$ be the substitution obtained by composition of $\theta_1$ and $\theta_2$. Let us prove that $\theta$ is a substitution such that $C$ is more general than $E$ under generalized implication. By definition of compound substitution, we can say that $head(C)\theta = head(C)\theta_1\theta_2 = head(D)\theta_2 = head(E)$. Given that the implication is monotone with respect to the application of a substitution, $T \models \forall(C\theta_1 \Rightarrow D)$ entails $T \models \forall(C\theta_1\theta_2 \Rightarrow D\theta_2)$. From $T \models \forall(C\theta_1\theta_2 \Rightarrow D\theta_2)$ rewritten as $T \models \forall(C\theta \Rightarrow D\theta_2)$ and $T \models \forall(D\theta_2 \Rightarrow E)$, $T \models \forall(C\theta \Rightarrow E)$ follows.

It can also be proved the semi-decidability of the generalized implication, namely the termination of the generalized implication test is guaranteed when $C \leq_{T,\Rightarrow} D$. However, such a negative result is overcome when Datalog clauses [6] are considered. In fact, the restriction to function-free clauses is common in ILP systems, which remove function symbols from clauses and put them in the background knowledge by techniques such as flattening [29].

The generalization model represents another difference between ATRE and MPL, whose refinement operators are based on $\theta$-subsumption. MPL adopts two different generalization models during its search: $\theta$-subsumption while learning a single clause, and logical entailment while learning the whole theory. Therefore, two distinct checks are performed by the system for each learned clause: a local consistency/completeness check based on $\theta$-subsumption and a global check based on logical entailment.

### 4.1 An Implementation of the Generalized Implication Test

A naive implementation of the generalized implication test is obtained by computing the least Herbrand models of $\{C\theta\} \cup T$ and $\{D\} \cup T$: $C \leq_{T,\Rightarrow} D$ if and only if $LHM(\{D\} \cup T) \subseteq LHM(\{C\theta\} \cup T)$ for some $\theta$ such that $head(C)\theta = head(D)$. Since the least Herbrand model of a theory $T$ coincides with the least fixed point of the immediate consequence operator $\tau_T$[2] we have an operative way to compute the least

[2] The standard notation used in logic programming and deductive databases for the immediate consequence operator is $T_P$, where $P$ is the logic program or the Datalog program.

Herbrand models. Moreover, the convergence to the fixpoint is guaranteed by the finiteness of the Herbrand base for Datalog theories.

One reason for inefficiency in the naive evaluation is that ground facts in $LHM(T)$ may be computed many times during the iterative application of $\tau_T$. Semi-naive evaluation partially overcomes this redundancy by partitioning $T$ into *n layers* such that $T=T^0\cup\ldots\cup T^i\cup\ldots\cup T^{n-1}$ and $LHM(T)= LHM(LHM(\cup_{j=0,\ldots,n-2}\ T^j)\cup T^{n-1})$.

It is worthwhile to notice that the computation of $LHM(LHM(\cup_{j=0,\ldots,i-1}\ T^j)\cup T^i)$ is equivalent to the iterative application of the immediate consequence operator to $T^i$ starting from the interpretation $LHM(\cup_{j=0,\ldots,i-1}\ T^j)$, that is $\tau_{T^i}\ (LHM(\cup_{j=0,\ldots,i-1}\ T^j))$. In this way, clauses in $T^0\cup\ldots\cup T^{i-1}$ are not considered anymore while computing the logical consequences of $T^i$.

**Example** Let $T'$ be the following theory:

$C_1$: $p(X) \leftarrow q(X)$

$C_2$: $q(Y) \leftarrow r(X,Y)$

and

$BK$: $\{q(a), r(a,b), s(b), r(c,b)\}$

Let us suppose that the theory $T=BK\cup T'$ may be partitioned as follows: $T=T^0\cup T^1\cup T^2$ where $T^0=\{r(a,b), s(b), r(c,b)\}$, $T^1=\{q(a), C_2\}$ and $T^2=\{C_1\}$.

Indeed, $LHM(T)= \{q(a), r(a,b), s(b), r(c,b), q(b), p(a), p(b)\}$, and

$LHM(T^0)= T^0$

$LHM(T^0\cup T^1)=LHM(LHM(T^0)\cup T^1)=LHM(T^0\cup T^1)=$

$= \{r(a,b),s(b),r(c,b),q(a),q(b)\}= \tau_{T^1}\ (\{r(a,b), s(b), r(c,b)\})= \tau_{T^1}\ (LHM(T^0))$

$LHM(T^0\cup T^1\cup T^2)=LHM(LHM(T^0\cup T^1)\cup T^2)=LHM(\{r(a,b),s(b),r(c,b),q(a),q(b)\}\cup\{C_1\}=$

$=\{r(a,b),s(b),r(c,b),q(a),q(b),p(a),p(b)\}=$

$= \tau_{T^2}\ (\{r(a,b),s(b),r(c,b),q(a),q(b)\})= \tau_{T^2}\ (LHM(T^0\cup T^1))$

Notice that according to the classical iterative application of the immediate consequence operator the ground atoms $p(a)$ and $q(b)$ would be computed both in $\tau_T\uparrow 2$ and $\tau_T\uparrow 3$, since:

$\tau_T\uparrow 0 := \varnothing$

$\tau_T\uparrow 1 := \tau_T(\tau_T\uparrow 0) = BK$

$\tau_T\uparrow 2 := \tau_T(\tau_T\uparrow 1) = BK\cup\{p(a), q(b)\}$

$\tau_T\uparrow 3 := \tau_T(\tau_T\uparrow 2) = BK\cup\{p(a), q(b)\}\cup\{ p(a), q(b), p(b)\}$

Issues related to the problem of finding layers of a recursive theory $T$ such that $LHM(T)=LHM(\cup_{j=0,\ldots,n-1}T^j)=LHM(LHM(\cup_{j=0,\ldots,n-2}T^j)\cup T^{n-1})$ are to be addressed. Difficulties arise because the dependency graph $\gamma(T)$ is a directed cyclic graph. In order to remove cycles from $\gamma(T)$ we resort to the notion of *strongly connected component* of a directed graph [19].

**Definition 6 (collapsed dependency graph).** Let $\gamma(T)$ be the dependency graph of a logical theory $T$. The *collapsed dependency graph* of $T$, denoted as $\hat{\gamma}(T)$, is a

---

Henceforth, the operator will be denoted by $\tau_T$ in order to avoid confusion with the symbol $T$ used for logical theories.

directed acyclic graph (dag) obtained by collapsing each (maximal) strongly connected component of $\gamma(T)$ into a single node.

Given the properties of *dag*'s, it is easy to compute the level of a predicate $p \in \pi(T)$ as the maximum distance of $[p]$ from a terminal node in $\hat{\gamma}(T)$ where terminal nodes are nodes with no out-coming edges.

**Definition 7 (predicate level).** Let $\hat{\gamma}(T)$ be the collapsed dependency graph of a logical theory $T$. The level of a predicate $p \in \pi(T)$ is given by:

$level(p) = 0$ if $[p]$ is a terminal node in $\hat{\gamma}(T)$,

$1 + \max \{level(q) \mid q \in \pi(T)$ and $[q]$ is a child of $[p]$ in $\hat{\gamma}(T) \}$ otherwise

Any logical theory can be layered on the basis of the level of its predicates.

**Definition 8 (layered theory).** Let $\hat{\gamma}(T)$ be the collapsed dependency graph of a logical theory $T$. Then $T$ can be partitioned into $n$ disjoint sets of clauses $T=T^0 \cup \ldots \cup T^i \cup \ldots \cup T^{n-1}$, called *layers*, such that

$$\forall\, i \in \{0, .., n\text{-}1\}: \pi(T^i) = \{p \in \pi(T) \mid level(p)=i\}.$$

It is worthwhile to observe that such technique for the layering of a logical theory induces a total order on the layers, $T^0 \leq \ldots \leq T^i \leq \ldots \leq T^{n-1}$.

**Example** Let $T$ be a theory obtained by the union of a background knowledge

$BK$: $\{f(a), s(a,b), s(b,c), s(c,d), s(d,e)\}$

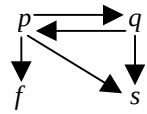
and a theory $T'$ consisting of

$C_1$: $p(X) \leftarrow f(X)$

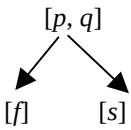
$C_2$: $q(Y) \leftarrow p(Z), s(Z,Y)$

$C_3$: $p(U) \leftarrow q(V), s(V,U)$

Given $T = BK \cup T'$, then $\pi(T)=\{f, s, p, q\}$ while $\gamma(T)$ is the following:




Thus $\hat{\pi}(T) = \{[f], [s], [p, q]\}$, and $\hat{\gamma}(T)$ is the following graph:




from which the two layers $T^0=BK$ and $T^1=T'$ are extracted.

The following proposition can be proved.

**Proposition 2.** *Let $H$ be a ground atom with predicate symbol $p \in \pi(T^k)$. Then $H \in LHM(T)$ if and only if $H \in LHM(LHM(\cup_{r=1,\ldots,k-1} T^r) \cup T^k)$.*

***Proof***

($\Rightarrow$) By induction over the layer $k$.

$k$=0. Let $H \in LHM(T)$. Then by the fix-point theorem, $\exists i>0$ such that $H \in \tau_T \uparrow i$, that is, by definition of immediate consequence operator, there exists a ground clause of $T$, $H \leftarrow A_1,\ldots,A_m \in ground(T)$, where $A_j \in \tau_T \uparrow (i\text{-}1)$, $j=1,2,\ldots,m$. Since $p \in \pi(T^0)$ then

$H \leftarrow A_1,\ldots,A_m \in ground(T^0)$. We want to prove that each $A_j \in LHM(T^0)$, from which $H \in LHM(T^0)$ follows.

The proof continues by induction over the iteration step $i$.

*i=1.* Since $H \in \tau_T \uparrow 1$ then $H \in T$, that is $H$ is a ground fact of the theory $T$. More specifically, $H \in T^0$, therefore $H$ is in the $LHM(T^0)$.

*i>1.* Each $A_j \in \tau_T \uparrow (i\text{-}1)$, therefore $A_j \in \mathrm{LHM}(T)$. From the construction of the layers follows that each $A_j$ is a ground atom with predicate symbol $p_j \in \pi(T^0)$. By the induction hypothesis each $A_j \in LHM(T^0)$, $j$=1,2, …, $m$.

$k$>0. Again, let $H \in LHM(T)$. Then for the fix-point theorem, $\exists i$>0 such that $H \in \tau_T \uparrow i$, that is, by definition of immediate consequence operator, there exists a ground clause of $T$, $H \leftarrow A_1,\ldots,A_m \in ground(T)$, where $A_j \in \tau_T \uparrow (i\text{-}1)$, $j$=1, 2, …,$m$. We want to prove that $A_j \in LHM(LHM(\cup_{r=1,\ldots,k\text{-}1} T^r) \cup T^k)$, from which $H \in LHM(LHM(\cup_{r=1,\ldots,k\text{-}1} T^r) \cup T^k)$ follows.

The proof continues by induction over the step $i$.

$i$=1. Since $H \in \tau_T \uparrow 1$ then $H \in T$, that is $H$ is a ground fact of the theory $T$. More specifically, $H \in T^k$, therefore $H$ is in the $LHM(T^k)$, and more in general $H$ is in $LHM(LHM(\cup_{i=1,\ldots,k\text{-}1} T^i) \cup T^k)$, since the addition of a set of clauses (i.e., the ground clauses $LHM(\cup_{i=1,\ldots,k\text{-}1} T^i)$) to a theory $T^k$ can only augment the least Herbrand model of the theory.

$i$>1. From the construction of the layers follows that each $A_j$ is a ground atom with predicate symbol $p_j \in \pi(T^r)$, with $r \le k$. Moreover $A_j \in \tau_T \uparrow (i\text{-}1)$, therefore $A_j \in LHM(T)$. By both inductive hypotheses (over $k$ and $i$), we may say that $A_j \in LHM(LHM(\cup_{r=1,\ldots,k\text{-}1} T^r) \cup T^k)$, for each $j$=1, 2, …,$m$, and then conclude that $H \in LHM(LHM(\cup_{r=1,\ldots,k\text{-}1} T^r) \cup T^k)$.

($\Leftarrow$) Let $H \in LHM(LHM(\cup_{r=1,\ldots,k\text{-}1} T^r) \cup T^k)$. In particular, if $H \in LHM(\cup_{r=1,\ldots,k\text{-}1} T^r)$ then $H \in LHM(T)$, since $\cup_{r=1,\ldots,k\text{-}1} T^r \subseteq T$. Otherwise, $H$ has been obtained by applying iteratively the immediate consequence operator $\tau_{T^k}$ starting from the interpretation $LHM(\cup_{r=1,\ldots,k\text{-}1} T^r) \subseteq LHM(T)$, that is $H \in \tau_{T^k}(\mathrm{LHM}(\cup_{r=1,\ldots,k\text{-}1} T^r)) \uparrow \infty$. Since $\tau_{T^k}$ is monotone and $T^k \subseteq T$, $\tau_{T^k}(LHM(\cup_{r=1,\ldots,k\text{-}1} T^r)) \uparrow \infty \subseteq \tau_{T^k}(LHM(T)) \uparrow \infty = LHM(T)$. Therefore $H \in LHM(T)$. ■

This proposition states that a necessary and sufficient condition for a ground atom with predicate symbol $p \in \pi(T^k)$ being in $LHM(T)$ is that $H$ is computed by iteratively applying the immediate consequence operator $\tau_{T^k}$ starting from the interpretation $LHM(\cup_{r=1,\ldots,k\text{-}1} T^r)$. Proposition 2 can be used to prove the following theorem on the least Herbrand model of a layered theory.

**Theorem.** *Let T be a theory which has been partitioned into n layers according to the criterion defined in Definition 8. Then*

$$\forall\, n \ge 1: LHM(T) = LHM(\cup_{r=0,\ldots,n\text{-}1} T^r) = LHM(LHM(\cup_{r=0,\ldots,n\text{-}2} T^r) \cup T^{n\text{-}1}).$$

***Proof***

($\Rightarrow$) Let $P^0$, $P^1$,…, $P^{n\text{-}1}$ be a partition of $LHM(T)$ such that each $P^i$, $i$=0, …, $n$-1, contains only ground atoms with some predicate symbol in $\pi(T^i)$. From Proposition 2 it follows $P^i \subseteq LHM(LHM(\cup_{r=1,\ldots,i\text{-}1} T^r) \cup T^i)$ and $LHM(T) = P^0 \cup P^1 \cup \ldots \cup P^{n\text{-}1} \subseteq LHM(T^0) \cup$

$LHM(LHM(T^0)\cup T^1)\cup\ldots\cup LHM(LHM(\cup_{r=0,\ldots,n\text{-}2} T^r)\cup T^{n\text{-}1})=LHM(LHM(\cup_{r=0,\ldots,n\text{-}2} T^r)\cup T^{n\text{-}1})$ since $LHM(\cup_{r=0,\ldots,i\text{-}1} T^r)\subseteq LHM(LHM(\cup_{r=0,\ldots,i\text{-}1} T^r)\cup T^i)$.

($\Leftarrow$) Let $H$ be in $LHM(LHM(\cup_{r=0,\ldots,n\text{-}2} T^r)\cup T^{n\text{-}1})$. If $H\in LHM(\cup_{r=0,\ldots,n\text{-}2} T^r)$ then $H\in LHM(T)$, since $\cup_{r=0,\ldots,n\text{-}2} T^r \subseteq T$. Otherwise, $H$ has been obtained by applying iteratively the immediate consequence operator $\tau_{T^{n-1}}$ starting from the interpretation $LHM(\cup_{r=1,\ldots,n\text{-}2} T^r)\subseteq LHM(T)$, that is $H\in \tau_{T^{n-1}}(LHM(\cup_{r=1,\ldots,n\text{-}2} T^r))\uparrow\infty$. Since $\tau_{T^{n-1}}$ is monotone and $T^{n-1}\subseteq T$, $\tau_{T^{n-1}}(LHM(\cup_{r=1,\ldots,n\text{-}2} T^r))\uparrow\infty \subseteq \tau_{T^{n-1}}(LHM(T))\uparrow\infty = LHM(T)$. Therefore $H\in LHM(T)$. ■

To sum up, the layering of a theory provides a semi-naive way of computing the generalized implication test presented above. The importance of layering will be more evident when the problem of recovering consistency will be faced (see the following Section).

## 5 The Consistency Recovery Strategy

Another learning issue to be considered in the multiple predicate learning is the non-monotonicity of the normal ILP setting: Whenever two individual clauses are consistent on the data, their conjunction needs not to be consistent on the same data [11]. Algorithmic implications of this property may be effectively illustrated by means of an example.

**Example.** Let the following sets be positive examples, negative examples and background knowledge respectively:

+: $\{p(a),\ p(c), p(e), q(b)\}$
—: $\{q(d)\}$
*BK*: $\{f(a), s(a,b), s(b,c), s(c,d), s(d,e)\}$

Let us suppose that the following consistent, but not complete, recursive theory $T_2$ has been learned after two conquer stages:

$C_1$: $p(X) \leftarrow f(X)$
$C_2$: $q(Y) \leftarrow p(Z), s(Z,Y)$

Note that $C_1 \leq_{\{C2\}\cup BK,\Rightarrow} \{p(a)\}$, and $C_2 \leq_{\{C1\}\cup BK,\Rightarrow} \{q(b)\}$, that is $T_2$ entails $p(a)$ and $q(b)$ given *BK*. Since $T_2$ is incomplete, the learner will generate a new clause, say

$C$: $p(U) \leftarrow q(V), s(V,U)$

which is consistent (it covers $p(c)$ alone, given $T_2\cup BK$), but when added to the recursive theory, it makes clause $C_2$ inconsistent ($C_2 \leq_{\{C1,C\}\cup BK,\Rightarrow} \{q(d)\}$).

There are several ways to remove such inconsistency by revising the learned theory. Nienhuys-Cheng and de Wolf [25] describe a complete method to specialize a logic theory with respect to sets of positive and negative examples. The method is based upon unfolding, clause deletion and subsumption. These operations are not applied to the last clause added to the theory, but may involve any clause of the inconsistent theory. As a result, clauses learned in the first inductive steps could be totally changed or even removed. This theory revision approach, however, is not coherent with the stepwise construction of the theory $T$ presented in Section 3, since it

re-opens the whole question on the validity of clauses added in the previous steps. An alternative approach consists of simple syntactic changes of the theory, whose ultimate effect is that of creating new *layers* in a logical theory, just as the stratification of a normal program creates new strata [2].

Our proposal of recovery strategy fits the latter approach since it is based on the layering technique illustrated in Section 4.1 (see Definition 8).

**Example** In the previous example, it is possible to define only two layers for $T_2 \cup BK$

1. $T^0=BK$ with $\pi(T^0)=\{f, s\}$, and
2. $T^1=\{C_1, C_2, C\}$ with $\pi(T^1)=\{p, q\}$.

By reformulating $C_1$ and $C_2$ as follows:

$C'_1$: $p'(X) \leftarrow f(X)$

$C'_2$: $q(Y) \leftarrow p'(Z), s(Z,Y)$

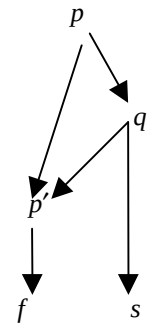
and by adding the following two clauses:

$C_3$: $p(W) \leftarrow p'(W)$

$C$: $p(U) \leftarrow q(V), s(V,U)$

the new theory $T'_2$ will present four layers:

1. $T^0=BK$ with $\pi(T^0)=\{f, s\}$,
2. $T^1=\{C'_1\}$ with $\pi(T^1)=\{p'\}$,
3. $T^2=\{C'_2\}$ with $\pi(T^2)=\{q\}$, and
4. $T^3=\{C_3, C\}$ with $\pi(T^3)=\{p\}$.




It is straightforward to notice that the theory $T'_2$ is consistent. As effect of theory restructuring, $p$ and $q$ are no longer in the same equivalence class in the collapsed dependency graph, and the number of layers increased.

More generally, let $T=T^0 \cup \ldots \cup T^i \cup \ldots \cup T^{n-1}$ and suppose that the addition of a clause $C$ to the theory $T$ makes a clause in $T^i$ inconsistent. The recovery strategy based on layering simply substitutes all occurrences in $T$ of the predicate $p$ in $head(C)$ with a new predicate symbol, $p'$, before adding the two clauses $C$ and $p(t_1,\ldots,t_n) \leftarrow p'(t_1,\ldots,t_n)$.

**Proposition 3**. *The new theory T′ obtained as above has a number of layers greater than or equal to T.*

***Proof***. Let $l$ be the $level(p)$ in $\hat{\gamma}(T)$. Since $p'$ replaces $p$ in $T$ it has the same level of $p$ before theory restructuring, namely $level(p')=l$ in $\hat{\gamma}(T')$. Moreover $level(p)>level(p')$ in $\hat{\gamma}(T')$. If $l$ equals the maximum level of a node in $\hat{\gamma}(T)$, then the new theory $T'$ has a predicate at a greater level, that is the number of layers in $T'$ increases. Conversely, the level of all predicates $q$ depending on $p$ in $T$ can either increase because of the breaking of equivalence classes or remain stable. ■

This proposition generalizes the consideration on the increase of layers made for the example above. The main problem remains the coverage of the new theory $T'$.

**Proposition 4**. *The new theory T′ is consistent.*

***Proof***. Let $T'' = T' \setminus \{C\}$. From the construction of $T''$ follows $LHM(T'') = LHM(T) \cup \{p'(t_1,\ldots,t_n) \mid p(t_1, \ldots,t_n) \in LHM(T)\}$. Indeed, $T''$ simply renames $p$ with $p'$ and adds the clause $p(t_1, \ldots,t_n) \leftarrow p'(t_1, \ldots,t_n)$. $T''$ is consistent with respect to the training set, since $LHM(T)$ does not include negative examples ($T$ is consistent before adding

$C$), and no conflict can be generated between the new ground atoms $p'(t_1,\ldots,t_n)$ and the training set. Moreover, no clause in $T''$ depends on $p$, because of renaming. Therefore, $LHM(T') = LHM(LHM(T'') \cup \{C\})$. Suppose that $T'$ is inconsistent, that is there exists a ground atom $H \in LHM(T')$ such that $H$ is a negative example in the training set. Obviously, $H \notin LHM(T'')$ since we have just proved $T''$ being consistent. By hypothesis, the clause $C$ is consistent given $T$ (otherwise it would not be selected during the learning process), therefore $LHM(LHM(T) \cup \{C\})$ does not contain negative examples. Since $LHM(T'') = LHM(T) \cup \{p'(t_1,\ldots,t_n) \mid p(t_1, \ldots,t_n) \in LHM(T)\}$ and $p'(t_1,\ldots,t_n)$ do not affect the set of consequences computed by the immediate consequence operator, we conclude that also $LHM(LHM(T'') \cup \{C\})$ does not contain negative examples. ■

**Proposition 5**. $LHM(T) \subseteq LHM(T')$.

***Proof***. Let $T'' = T' \setminus \{C\}$. By construction of $T'$ and $T''$, the thesis follows from the chaining of the inclusions $LHM(T') = LHM(LHM(T'') \cup \{C\}) \supseteq LHM(T'')$ and $LHM(T'') = LHM(T) \cup \{p'(t_1,\ldots,t_n) \mid p(t_1,\ldots,t_n) \in LHM(T)\} \supseteq LHM(T)$. ■

To sum up, the new theory $T'$ is consistent and keeps the original coverage of $T$.

It is noteworthy that in the proposed approach to consistency recovery new predicates are invented, aiming at accommodating previously acquired knowledge (theory) with the currently generated hypothesis (clause). This approach differs from that adopted by MPL, which recovers inconsistency by allowing removal of globally inconsistent clauses from the current theory.

## 6 Application to the Document Understanding Problem

The proposed approach to multiple predicate learning has been implemented in the learning system ATRE, whose representation language can be easily transformed into Datalog clauses extended with built-in predicates. An application to the problem of learning the mutual recursive predicate definition of *even* and *odd* is reported in [21]. ATRE has been also applied to the problem of processing printed documents and its induced logical theories are used by an intelligent document processing system, named WISDOM++ (http://www.di.uniba.it/~malerba/wisdom++/) [13]. Henceforth, only the specific problem of learning rules for document understanding will be addressed. The main novelty with respect to previous work is the automated discover of possible concept dependencies as well as the consideration of multi-page documents.

A document is characterized by two different structures representing both its internal organization and its content: The *layout* (or *geometrical*) structure and the *logical* structure. The former associates the content of a document with a hierarchy of layout objects, such as text lines, vertical/horizontal lines, graphic/photographic elements, pages, and so on. The latter associates the content of a document with a hierarchy of logical objects, such as sender/receiver of a business letter, title/authors of an article, and so on. Here, the term document understanding denotes the process of mapping the layout structure of a document into the corresponding logical structure. The document understanding process is based on the assumption that documents can be understood by means of their layout structures alone.

The mapping of the layout structure into the logical structure can be represented as a set of rules. Traditionally, such rules have been hand-coded for particular kinds of document [24], requiring much human tune and effort. We propose the application of inductive learning techniques in order to generate automatically the rules from a set of training examples. The user-trainer is asked to label some layout components of a set of training documents according to their logical meaning. Those layout components with no clear logical meaning are not labeled. Therefore, each document generates as many training examples as the number of layout components. Classes of training examples correspond to the distinct logical components to be recognized in a document. The unlabelled layout objects play the role of counterexamples for all the classes to be learned.

Each training example is represented as an *object* in ATRE, namely a multiple-head ground clause, where different constants represent distinct layout components of a page layout. The description of a document page is reported in Fig. 3 while Table 1 lists all descriptors used to represent a page layout of a multi-page document.

**Table 1.** Descriptors used by WISDOM++ to represent a page layout of a multi-page document.

| ***Descriptor*** | ***Domain*** |
|---|---|
| page(page) | *Nominal domain:* first, intermediate, last_but_one, last |
| width(block) | *Integer domain:* (1..640) |
| height(block) | *Integer domain:* (1..875) |
| x_pos_centre(block) | *Integer domain:* (1..640) |
| y_pos_centre(block) | *Integer domain:* (1..875) |
| type_of(block) | *Nominal domain:* text, hor_line, image, ver_line, graphic, mixed |
| part_of(page,block) | *Boolean domain:* true if page contains block |
| on_top(block1,block2) | *Boolean domain:* true if block1 is above block2 |
| to_right(block1,block2) | *Boolean domain:* true if block2 is to the right of block1 |
| alignment(block1,block2) | *Nominal domain:* only_left_col, only_right_col, only_middle_col, both_columns, only_upper_row, only_lower_row, only_middle_row, both_rows |

The following rules are used as background knowledge, in order to automatically associate information on page order to layout blocks.

*at_page(X)=first ← part_of(Y,X), page(Y)=first*
*at_page(X)=intermediate← part_of(Y,X), page(Y)=intermediate*
*at_page(X)=last_but_one ← part_of(Y,X), page(Y)=last_but_one*
*at_page(X)=last ← part_of(Y,X), page(Y)=last*

Three long papers appeared in the January 1996 issue of the IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI) have been considered. The papers contain thirty-seven pages, each of which has a variable number of layout components (about ten on average). Layout components can be associated with at most one of the

following eleven logical labels: *abstract*, *affiliation*, *author*, *biography*, *caption*, *figure*, *index_term*, *page_number*, *references*, *running_head*, *title.*

Learning rules for the recognition of semantically relevant layout components in a document raises issues concerning the induction of recursive theories. Simple and mutual concept dependencies are to be handled, since the logical components refer to a part of the document rather than to the whole document and may be related to each other. For instance, in case of papers published in journals, the following dependent clauses:

*running_head(X)* ← *top_left (X), text(X), even_page_number(X)*
*running_head(X)* ← *top_right (X), text(X), odd_page_number(X)*
*paragraph(Y)* ← *ontop(X,Y), running_head(X), text(Y)*

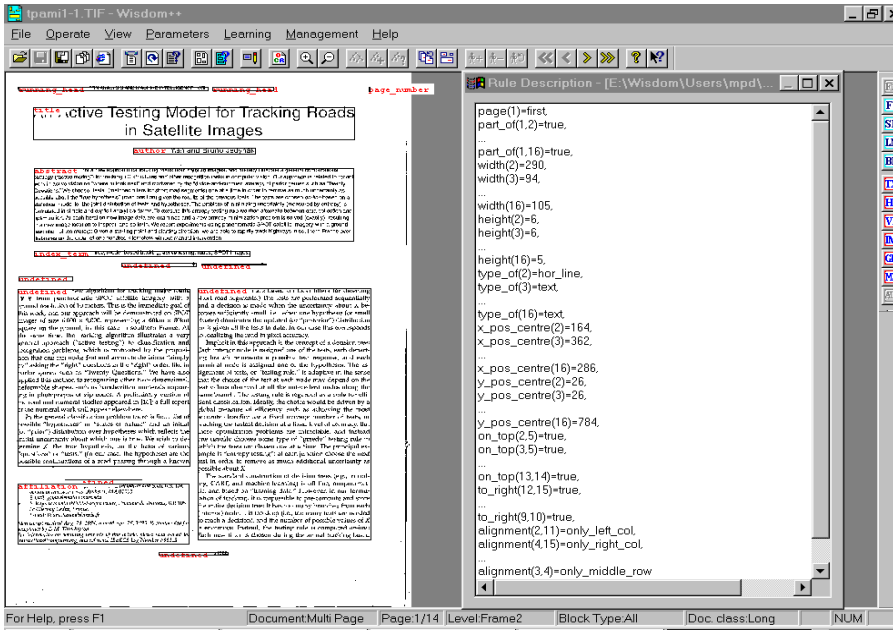



**Fig. 3.** Layout of the first page of a multi-page document (left) and its partial description in a first-order logic language (right).

express the fact that a textual layout component at the top left (right) hand corner of an even (odd) page is a running head, while a textual layout component below a running-head is a paragraph of the paper. Moreover, the recursive clause

*paragraph(Y)* ← *ontop(X,Y), paragraph(X), text(Y)*

is useful to classify all textual layout components below the upper-most paragraph. Therefore, document understanding seems to be the kind of application that may benefit of learning strategies for multiple predicate learning. Generally, the main benefits are:

- *Learnability of correct concept definitions*. For instance, some relational systems that do not take concept dependencies into account such as FOIL [28], cannot

learn the definitions of "appending two lists" and "reversing a list" independently, since the former concept is essential to give a reasonably compact definition of the second concept.

- *Rendering explicit some concept dependencies*, which would be otherwise hidden in a set of flat, independent rules. A correct logical theory structured around a number of dependent concepts does not contain those redundancies of its equivalent theory with independent concepts, therefore it is more comprehensible and easier to be validated by experts.

By running ATRE on the training set described above, the following theory is returned:

1. *logic_type(X)= running_head ← y_pos_centre(X)∈[18 .. 39], width(X)∈[77 .. 544]*
2. *logic_type(X)= page_number ← width(X) ∈ [2 .. 8], y_pos_centre(X)∈[19 .. 40]*
3. *logic_type(X)= figure ← type_of(X)=image, at_page(X)=intermediate*
4. *logic_type(X)= figure ← type_of(X)=graphic*
5. *logic_type(X)= abstract ← at_page(X)=first, width(X)∈[487 .. 488]*
6. *logic_type(X)= affiliation ← at_page(X)=first, y_pos_centre(X)∈[720 .. 745]*
7. *logic_type(X)= caption ← height(X)∈[9 .. 75], alignment(Y,X)=only_middle_col, logic_type(Y)= figure, type_of(X)=text*
8. *logic_type(X)= author ← at_page(X)=first, y_pos_centre(X)∈[128 .. 158]*
9. *logic_type(X) = references ← height(X)∈[332 .. 355], x_pos_centre(X)∈[153 .. 435]*
10. *logic_type(X) = title ← at_page(X)=first, height(X)∈[18 .. 53]*
11. *logic_type(X)=biography ← at_page(X)=last, height(X)∈[65 .. 234]*
12. *logic_type(X)=caption ← height(X)∈[9 .. 75], on_top(Y,X) , logic_type(Y)=figure, type_of(X)=text, to_right(Z,Y)*
13. *logic_type(X)=index_term ← height(X)∈[8 .. 8], y_pos_centre(X)∈[263 .. 295]*
14. *logic_type(X)=caption ← alignment(X,Y)=only_lower_row, height(X)∈[9 .. 9]*
15. *logic_type(X)=caption ← on_top(Y,X), logic_type(Y)=figure, type_of(X)=text, alignment(Y,Z)=only_right_col*
16. *logic_type(X) = caption ← height(X)∈[9 .. 75], on_top(X,Y), logic_type(Y)=figure, type_of(X)=text, type_of(Y)=graphic*
17. *logic_type(X)=caption ← height(X)∈[9 .. 75], alignment(Y,X)=only_left_col, alignment(Z,Y)=only_left_col, logic_type(Z)=caption, width(Z)∈[467 .. 546]*

Clauses are reported in the order in which they are learned. They have been filtered out from candidate clauses during the evaluation step performed according to a composite preference criterion that minimizes the number of negative examples covered, maximizes the number of positive examples covered, and minimizes the cost of the clause. The theory contains some concept dependencies (see clauses 7 and 12) as well as some kind of recursion (se clause 17). Learned concepts belong to two distinct layers: *abstract*, *affiliation*, *author*, *biography*, *figure*, *index_term*, *page_number*, *references*, *running_head*, and *title* are in one layer, while *caption* in the other. During the learning process, it was not necessary to recover theory consistency, and from our experience, the cases in which the recovery is required are rare. Surprisingly, some expected concept dependencies were not discovered by the system, such as that relating the running head to the page number:

*logic_type(X)= page_number ← to_right(X,Y), logic_type(Y)= running_head*
*logic_type(X)= page_number ← to_right(Y,X), logic_type(Y)= running_head*

The reason is due to the semantics of the descriptor *to_right*, which is generated by WISDOM++ only when two layout components are at a maximum distance of 100 points, which is not the case of articles published on the PAMI transactions. Same consideration applies to other possible concept dependencies (e.g., title-authors-abstract).

In order to test the predictive accuracy of the learned theory, we considered the fourth long article published in the same issue of the transactions used for training. WISDOM++ segmented the fourteen pages of the article into 169 layout components, sixteen of which (i.e., less than 10%) could not be properly labeled using by the learned theory (omission errors). No commission error was observed. This is important in this application domain, since commission errors can lead to totally erroneous storing of information. Finally, it is important to observe that many omission errors are due to near misses. For instance, the running head of the first page is not recognized simply because its centroid is located at point 40 along the vertical axis, while the range of *y_pos_center* values determined by ATRE in the training phase is [18..39] (see clause 1). Significant recovery of omission errors can be obtained by relaxing the definition of flexible matching between definite clauses [12].

## 7 Conclusions and Future Work

In this paper we have discussed and proposed computational solutions to some relevant issues raised by the induction of recursive theories in the normal ILP setting. A separate-and-parallel-conquer search strategy has been adopted to synchronize and interleave the learning of clauses supplying predicates with mutually recursive definitions. A novel generality order, called generalized implication, has been imposed to the search space of clauses in order to cope with recursion in a more suitable way. A layering technique based on the collapsed dependency graph has been investigated to recover the consistency of a partially learned theory. These ideas have been implemented in the ILP system ATRE and tested in the specific context of the document understanding problem. Experimental results obtained on a set of real-world multi-page documents empirically prove the validity of our approach to multiple predicate learning as well as the importance of taking predicate dependencies into account in the chosen domain application. In the future, we mean to refine the solutions being proposed in order to remove some inefficiency. In particular, it is necessary to optimize the separate-and-parallel-conquer search strategy with the aim of preventing it from exploring the specialization hierarchies repeatedly during the learning process. Moreover, we plan to investigate further the reasons causing the unsuccessful discovery of some expected concept dependencies. Finally, it is worth to test the system performance in case of document understanding problems with a richer set of logical components to be recognized.

## References


1. Aha, D.W., Lapointe, S., Ling, C.X., Matwin, S.: Learning recursive relations with randomly selected small training sets. Proc. 11th Int. Conf. on Machine Learning, (1994) 12-18
2. Apt, K.R.: Logic programming. In: van Leeuwen, J. (ed.): Handbook of Theoretical Computer Science, Vol. B. Elsevier, Amsterdam (1990) 493-574
3. Boström, H.: Specialization of recursive predicates. In Lavrac, N., Wrobel, S. (eds.): Machine Learning ECML-95. Lecture Notes in Artificial Intelligence, Vol. 912. Springer-Verlag, Berlin (1995) 92-106
4. Buntine, W.: Generalised subsumption and its applications to induction and redundancy. Artificial Intelligence, Vol. 36 (1988) 149-176
5. Cameron-Jones, R.M., Quinlan, J.R.: Avoiding pitfalls when learning recursive theories. Proc. 12th Int. Joint Conf. on Artificial Intelligence, (1993) 1050-1055
6. Ceri, S., Gottlob, G., Tanca, L.: What you always wanted to know about Datalog (and never dared to ask). IEEE Transactions on Knowledge and Data Engineering 1(1) (1989) 146-166
7. Cohen, W. W.: Learnability of restricted logic programs. In: Muggleton, S. (ed.): Proc. 3rd Int. Workshop on Inductive Logic Programming, (1993) 41-71
8. De Raedt, L.: Interactive Theory Revision. Academic Press, London (1992)
9. De Raedt, L., Lavrac, N., Dzeroski, S.: Multiple predicate learning. Proc. 13th Int. Joint Conf. on Artificial Intelligence, (1993) 1037-1042
10. De Raedt, L., Lavrac, N.: The many faces of inductive logic programming. In: Komorowski, J., Ras, Z.W. (eds.): Methodologies for Intelligent Systems. Lecture Notes in Artificial Intelligence, Vol. 689. Springer-Verlag, Berlin (1993) 435-449
11. De Raedt, L., Dehaspe, L.: Clausal discovery. Machine Learning 26(2/3), (1997) 99-146
12. Esposito, F., Caggese, S., Malerba, D., Semeraro, G.: Classification in noisy domains by flexible matching. Proc. European Symposium on Intelligent Techniques, (1997) 45-49
13. Esposito, F., Malerba, D., Lisi, F.A.: Machine Learning for Intelligent Processing of Printed Documents. In: Ras, Z.W., Skowron, A. (eds.): Journal of Intelligent Information Systems 16. Kluwer Academic Publishers (2000) 175-198
14. Flach, P.: A framework for inductive logic programming. In: Muggleton, S. (ed.): Inductive Logic Programming, Vol. 38 of Apic Series. Academic Press, London (1992) 193-211
15. Flener, P., Yilmaz, S.: Inductive Synthesis of Recursive Logic Programs: Achievements and Prospects. Journal of Logic Programming 41(2/3), Special Issue on Synthesis, Transformation and Analysis, (1999) 141-195
16. Giordana, A., Saitta, L., Baroglio, C.: Learning simple recursive theories. In: Komorowski, J., Ras, Z.W. (eds.): Methodologies for Intelligent Systems. Lecture Notes in Artificial Intelligence, Vol. 689. Springer-Verlag, Berlin (1993) 425-434
17. Idestam-Almquist, P.: Efficient induction of recursive definitions by structural analysis of saturations. In: De Raedt, L. (ed.): Advances in Inductive Logic Programming. IOS Press, Amsterdam (1996) 192-205
18. Jorge, A., Brazdil, P.: Architecture for iterative learning of recursive definitions. In: De Raedt, L. (ed.): Advances in Inductive Logic Programming. IOS Press, Amsterdam (1996) 206-218
19. van Leeuwen, J.: Graph Algorithms. In: van Leeuwen, J. (ed.): Handbook of Theoretical Computer Science, Vol. A. Elsevier, Amsterdam (1990) 525-631
20. Malerba, D., Semeraro, G., Esposito, F.: A multistrategy approach to learning multiple dependent concepts. In: Nakhaeizadeh, G., Taylor, C. (eds.): Machine Learning and Statistics: The interface. John Wiley & Sons, New York (1997) 87-106
21. Malerba, D., Esposito, F., Lisi, F.A.: Learning Recursive Theories with ATRE. In: Prade, H. (ed.), Proc. 13th Europ. Conf. on Artificial Intelligence. John Wiley & Sons, Chichester (1998) 435-439
22. Mitchell, T.M.: Machine Learning. McGraw-Hill (1997)

23. Mofizur, C.R., Numao, M.: Top-down induction of recursive programs from small number of sparse examples. In: De Raedt, L. (ed.): Advances in Inductive Logic Programming. IOS Press, Amsterdam (1996) 236-253
24. Nagy, G., Seth, S.C., Stoddard, S.D.: A prototype document image analysis system for technical journals. IEEE Computer 25(7), (1992) 10-22
25. Nienhuys-Cheng, S.-W., de Wolf, R.: A complete method for program specialization based upon unfolding. Proc. 12th Europ. Conf. on Artificial Intelligence (1996) 438-442
26. Plotkin, G.D.: A note on inductive generalization. In: Meltzer, B., Michie, D. (eds.): Machine Intelligence 5. Edinburgh University Press, Edinburgh (1970) 153-163
27. Plotkin, G.D.: A further note on inductive generalization. In: Meltzer, B., Michie, D. (eds.): Machine Intelligence 6. Edinburgh University Press, Edinburgh (1971) 101-124
28. Quinlan, J.R.: Learning Logical Definitions from Relations. Machine Learning 5 (1990) 239-266
29. Rouveirol, C.: Flattening and saturation: Two representation changes for generalization. Machine Learning 14(2) (1994) 219-232