On the LearnAbility of Abstraction Theories from Observations for Relational Learning

Stefano Ferilli, Teresa M.A. Basile,
Nicola Di Mauro, and Floriana Esposito

Department of Computer Science,
University of Bari, Italy
{ferilli, basile, ndm, esposito}@di.uniba.it

Abstract. The most common methodology in symbolic learning consists in inducing, given a set of observations, a general concept definition. It is widely known that the choice of the proper description language for a learning problem can affect the efficacy and effectiveness of the learning task. Furthermore, most real-world domains are affected by various kinds of imperfections in data, such as inappropriateness of the description language which does not contain/facilitate an exact representation of the target concept. To deal with such kind of situations, Machine Learning approaches moved from a framework exploiting a single inference mechanism, such as induction, towards one integrating multiple inference strategies such as abstraction. The literature so far assumed that the information needed to the learning systems to apply additional inference strategies is provided by a domain expert. The goal of this work is the automatic inference of such information.

The effectiveness of the proposed method was tested by providing the generated abstraction theories to the learning system INTHELEX as a background knowledge to exploit its abstraction capabilities. Various experiments were carried out on the real-world application domain of scientific paper documents, showing the validity of the approach.

1 Introduction

Although the efficacy of induction algorithms has been demonstrated on a wide variety of benchmark domains, current Machine Learning techniques are inadequate for more difficult real-world domains. The nature of the problem can be of different types, such as noise in the descriptions and lack of data but also the low level representation of the examples of the target concept. It is well known that the inappropriateness of a description language that does not contain/facilitate an exact representation of the target concept can affect the efficacy/effectiveness of the learning task. Hence, the choice of the proper representation for a learning problem has a significant impact on the performance, of Machine Learning systems in general, and of ILP systems [10] in particular. Generally, a low level representation provides all the information necessary to the learning task, but its individual parts are only remotely related to the target concept, making patterns

J. Gama et al. (Eds.): ECML 2005, LNAI 3720, pp. 120–132, 2005.

hard to identify. Low level representations are common in real-world domains, where examples are naturally described by many small measurements, in which there is not enough knowledge to represent the data with few highly relevant features.

Among various strategies proposed to overcome this limitation, there are different ways to exploit the *abstraction* framework proposed in [14]. For example, [16] addresses the problem of potentially many mappings that can hold between descriptions in a first-order representation language by selecting one particular type of mapping at a time and using it as a basis to define a new hypothesis space, thus performing a *representation change*. [15] used it to overcome the knowledge acquisition bottleneck that limits the learning task in particular application domains such as the automation of cartographic generalization. More generally, abstraction is used to model *a priori* the hypothesis space before the learning process starts introducing it as a multi-strategy capability that could shift to a higher language bias when the current one does not allow to capture the target predicate definition [3, 6, 8]. From an operational viewpoint, it should deal with cases in which learning can be more effective if it takes place at multiple (different) levels of complexity, which can be compared to the language bias shift considered in [2]; a useful perspective for the integration of this inference operator in an inductive learning framework was given in [14]. According to such a framework, the abstraction operator was endowed in the learning system INTHELEX [4] making it able to perform the shift.

In the current practice, it is in charge of the human expert to specify all the information needed by such a strategy for being applicable. It goes without saying that quality, correctness and completeness in the formalization of such information is a critical issue, that can determine the very feasibility of the learning process. Providing it is a very difficult task because it requires a deep knowledge of the application domain, and is in any case an error-prone activity, since omissions and errors may take place. For instance, the domain and/or the language used to represent it might be unknown to the experimenter, because he is just in charge of properly setting and running the learning system on a dataset provided by third parties and/or generated by other people. In any case, it is often not easy for non-experts to single out and formally express such knowledge in the form needed by the automatic systems, just because they are not familiar with the representation language and the related technical issues.

These considerations would make it highly desirable to develop procedures that automatically generate such information. This work aims at proposing solutions to automatically infer the information required by the abstraction framework from the same observations that are input to the inductive process, assuming that they are sufficiently significant, and at assessing the validity and performance of the corresponding procedures. In the following, after an introduction to the general framework for abstraction, the method for the automatic definition of appropriate rules to fire the operator will be presented along with an experimental session on a real-world domain.

2 Abstraction Inference Strategy: The General Framework

Abstraction is defined as a mapping between representations that are related to the same reference set but contain less detail (typically, only the information that is relevant to the achievement of the goal is maintained). It is useful in inductive learning when the current language bias proves not to be expressive enough for representing concept descriptions that can explain the examples, as discussed in [2].

Definition 1. *Given two clausal theories T (*ground theory*) and T' (*abstract theory*) built upon different languages $\mathcal{L}$ and $\mathcal{L}'$ (and derivation rules), an* abstraction *is a triple (T, T', f), where f is a computable total mapping between clauses in $\mathcal{L}$ and those in $\mathcal{L}'$.*

An Abstraction Theory (an operational representation of f) is used to perform such a *shift of language bias* [13, 2] to a higher level representation:

Definition 2. *An* abstraction theory *from $\mathcal{L}$ to $\mathcal{L}'$ is a consistent set of clauses $c : -d_1, \ldots, d_m$ where c is a literal built on predicates in $\mathcal{L}'$, and d_j, $j = 1, \ldots, m$ are literals built on predicates of $\mathcal{L}$. In other words, it is a collection of intermediate concepts represented as a disjunction of alternative definitions.*

Inverse resolution operators [10], by tracking back resolution steps, can suggest new salient properties and relations of the learning domain. Thus, they can be a valuable mechanism to build abstraction theories, as introduced in [7]. To this purpose, the absorption, inter-construction and intra-construction operators can be exploited, also in the case of first-order clauses. In this work we are interested in the case of a Datalog program [1, 9] as ground space of the abstraction, as in [11], where clauses are *flattened*, hence function-free.

Definition 3 (Absorption & Inter-construction).

absorption: *let C and D be Datalog clauses. If $\exists\theta$ unifier such that $\exists S \subset body(C)$, $S = body(D)\theta$, then applying the absorption operator yields the new clause C' such that:*

- *$head(C') = head(C)$*
- *$body(C') = (body(C) \setminus S) \cup \{head(D)\theta\}$,*

i.e., if all conditions in D are verified in the body of C, the corresponding literals are eliminated and replaced by $head(D)$.

Example 1. Let be C and D the following clauses:
C = bicycle(bb) ← has_pedals(bb,p), has_saddle(bb,s), has_frame(bb,f),
part_of(bb,w1), circular(w1), has_rim(w1), has_tire(w1),
part_of(bb,w2), circular(w2), has_rim(w2), has_tire(w2).
D = wheel(X) ← circular(X), has_rim(X), has_tire(X).
For such two clauses there exists $\theta_1 = X\backslash w1$ and $\theta_2 = X\backslash w2$, thus, applying absorption operator twice we obtain the following clause:
C' = bicycle(bb) ← has_pedals(bb,p), has_saddle(bb,s), has_frame(bb,f),
part_of(bb,w1), wheel(w1), part_of(bb,w2), wheel(w2).

inter-construction: *let* $C = \{C_i | i = 1, \ldots, n\}$ *be a set of Datalog clauses. If there exists a set of literals* R *and a unifier* θ_i *for each clause* C_i*, such that* $\exists S_i \subset body(C_i),\ S_i = R\theta_i$*, then we define:*

- *a new predicate* $L \leftarrow R$
- *for all* $i = 1, \ldots, n$ $body(C_i)$ *can be rewritten as* $(body(C_i) \setminus S_i) \cup \{L\theta_i\}$.

i.e., if all conditions in R *are verified in the body of each* $C_i \in C$*, the corresponding literals are eliminated and replaced by* L *that is a new predicate, with a definition in the theory, never present in the description language.*

Example 2. Let C *be the following set of clauses:*

C_1 = monocycle(m) ← has_small_pedals(m,sp), has_small_saddle(m,ss),
part_of(m,w1), circular(w1), has_rim(w1), has_tire(w1).

C_2 = bicycle(bi) ← has_pedals(bi,p), has_saddle(bi,s), has_frame(bi,f),
part_of(bi,wbi1),circular(wbi1),has_rim(wbi1),has_tire(wbi1),
part_of(bi,wbi2), circular(wbi2), has_rim(wbi2), has_tire(wbi2).

C_3 = car(c) ← has_motor_engine(c,me), has_steering_wheel(c,sw),
part_of(c,wc1),circular(wc1),has_rim(wc1),has_tire(wc1),
part_of(c,wc2),circular(wc2),has_rim(wc2),has_tire(wc2),
part_of(c,wc3),circular(wc3),has_rim(wc3),has_tire(wc3),
part_of(c,wc4),circular(wc4),has_rim(wc4),has_tire(wc4).

As we can note, the set $R = part\_of(A, B), circular(B), has\_rim(B), has\_tire(B)$ *is present in all the clauses and there exists an unifier between* R *and each of the clauses* C_1, C_2, C_3*, then it is possible to define a new predicate, let be it* $l(A, B)$*, and the clause* $l(A, B)\ :\ -part\_of(A, B), circular(B), has\_rim(B), has\_tire(B)$. *By this definition the set* C *can be rewritten as:*

C_1 = monocycle(m) ← has_small_pedals(m,sp), has_small_saddle(m,ss), l(m,w1).

C_2 = bicycle(bi) ← has_pedals(bi,p), has_saddle(bi,s), has_frame(bi,f),
l(bi,wbi1), l(bi,wbi2).

C_3 = car(c) ← has_motor_engine(c,me), has_steering_wheel(c,sw),
l(c,wc1), l(c,wc2), l(c,wc3), l(c,wc4).

In the framework for integrating abstraction and inductive learning given in [14], concept representation deals with entities belonging to three different levels, that together form a *reasoning context*. Underlying any source of experience is the *world*, where *concrete* objects (the 'real things') reside, that is not directly known, since any observer's access to it is mediated by his *perception* of it $P(W)$ (consisting of the 'physical' stimuli produced on the observer). To be available over time, these stimuli must be memorized in an organized *structure* S, i.e. an *extensional* representation of the perceived world, in which stimuli related to each other are stored together. Finally, to reason about the perceived world and communicate with other agents, a *language* L is needed, that describes it *intensionally*. Generally these sets contain operators for performing operations such as: grouping indistinguishable objects into equivalence classes; grouping a set of ground objects to form a new compound object that replaces them in the abstract world; ignoring terms, that disappear in the abstract world; merging a subset of values that are considered indistinguishable; dropping predicate arguments, thus reducing the arity of a relation (even to zero, thus moving to a propositional logic setting). Corresponding instances of these operators are present at each level of the reasoning context, so that it is possible to reason at any of the given levels.

3 Learning Abstraction Theories

The abstraction procedure reported in Section 2 aims at discarding or hiding the information that is insignificant to the achievement of the goal. According to Definitions 1 and 2, abstraction is based on a computable mapping f whose operational representation is an Abstraction Theory that encodes the abstraction operators by means of a consistent set of clauses, i.e. domain rules. Thus, in order to perform abstraction, an inductive concept learning system must be provided with an abstraction theory for the specific application domain at hand. As already pointed out, a common assumption is that such a knowledge is provided by an expert of the application domain. Here, we propose a general approach to automatically learn such a knowledge (domain rules) by looking for correspondences that often or seldom hold among a significant set of observations. These correspondences are generated according to the *inter-construction* operator (Definition 3) and are then exploited to simplify the description language in two different ways: by generating *shifting rules* that replace significant (characteristic or discriminant) groups of literals by one single literal representing their conjunction, or by generating *neglecting rules* that eliminate groups of literals that are not significant. Both kinds of rules will be applied in order to perform the shift of language bias according to the absorption operator presented in Definition 3 in this way reducing the description length and thus improving the induction performance.

Algorithm 1 sketches the overall procedure conceived to discover common paths in the application domain that potentially could make up the Abstraction Theory. It firstly generates domain rules involving unary predicates only, that represent the characteristics of an object in the description, and then the rules made up of predicates whose arity is greater than 1, that represent the relationships between two or more objects contained in the descriptions. The algorithm is based on the choice of an observation (referred to in the following as the *seed*) that will act as the representative of the concepts to be abstracted (currently it is the first encountered positive observation).

For each constant c_i in the seed description, the algorithm collects the unary predicates it is argument of, and computes all their subsets (excluding those having cardinality equal to 0, that do not give information about the object, or 1, that represent only properties of the objects). Each subset identified in this way is a candidate to compose the body of a rule, in the Abstraction Theory, made up of unary predicates. The selection among these subsets is done considering the ones that are the best representative for the class of the concept to be abstracted according to the seed e. Thus, each subset is assigned a score based on the number of times that it occurs in the positive and negative descriptions. This value represents the *coverage rate* of the subset with respect to the observations

Algorithm 1. Identification of domain rules for Abstraction Operators

```
Require: $\mathcal{E}^+$: set of positive observations; $\mathcal{E}^-$: set of negative observations; $e$: seed;
  Provide: $AT$: set of domain rules that make up an abstraction theory;
  if $\exists$ unary predicates in $e$ then
    $S := \emptyset$, $UnaryPreds :=$ set of unary predicates in $e$
    $C := \{c_1, c_2, \ldots, c_n\}$ set of constants in the description of $e$
    for all $c_i \in C$ do
      $S_i := \{l_i \in UnaryPreds$ s.t. $c_i$ is argument of $l_i\}$
      if ($\mid S_i \mid \neq 0$ and $\mid S_i \mid \neq 1$) then $S := S \bigcup S_i$
    for i=1..n do
      for all $S_j \in S$ do
        find all the subsets $s_{jm}$ of $S_j$ s.t.
          $(0 - \alpha \leq Score(s_{jm}) \leq 0 + \alpha)$ OR $(Max - \alpha \leq Score(s_{jm}) \leq Max + \alpha)$
        create the rule: $rule_{s_{jm}}(c_i) \leftarrow s_{jm}$
        replace in $\mathcal{E}^+$, in $\mathcal{E}^-$ and in $e$, $s_{jm}$ with $rule_{s_{jm}}(c_i)$
  while $F$ (:= set of all leaf predicates of $e$) $\neq \emptyset$ do
    for all $l_i \in F$ do
      if $l_i$ has only one parent (let $g_i(a_i, \ldots, a_n)$ be the $l_i$'s parent) then
        create the rule: $rule_{l_i}(a_i, ...a_n) \leftarrow g_i, l_i$; H := true
        replace in $\mathcal{E}^+$, in $\mathcal{E}^-$ and in $e$, $g_i, l_i$ with $rule_{l_i}(a_i, ...a_n)$)
    for all $rule_i \leftarrow l_{i_1}, \ldots, l_{i_n}$ generated do
      if $\{l_{i_1}, \ldots, l_{i_n}\}$ occurs in some rule $rule_j$ then
        replace $l_{i_1}, \ldots, l_{i_n}$ in $rule_j$ by $rule_i$
        eliminate $rule_i$ form the set of rules generated
  Evaluate the set of generated rules
```

and indicates the quality of the subset. This kind of selection allows to choose the subsets that are neither too specific, because they are present in few observations, nor too general, because they are encountered in almost all the observations. Once the subsets S_j are selected, the rules to make the Abstraction Theory are formulated in the following way:

$abstract\_predicate(c_i) \leftarrow S_j$ iff $score(S_j) \geq P$ (shifting rule)
$\leftarrow S_j$ iff $score(S_j) \leq P$ (neglecting rule)

where P is a threshold depending on the application domain at hand (in order to make P independent on the specific domain, the score can be normalized as a percentage of the maximum score actually computed in the given dataset). In the case of shifting rules, the rule's body S_j, that is a conjunction of literals, is very characterizing of either the positive or the negative observations, thus it is fundamental for the learning process and deserves to be identified by a specific predicate. In the case of neglecting rules, S_j could indicate a detail in the description that is not very significant for the learning process and thus it can be dropped. In both cases, replacing the rule's body with its head in the observations reduces the length of observations, this way making the learning process more efficient.

The algorithm continues with the identification of rules made up of predicates whose arity is greater than 1. Thus, once the previously identified abstraction rules are replaced in all the observations, they don't contain any unary predicates belonging to the original representation language. At this point, an iteration that groups together the n-ary predicates is performed until one of the following conditions succeeds: 1) the description of the seed e does not contain *leaf predicates* (predicates that share arguments with at least another predicate, excluding the head's predicate); 2) all the rules generated at step n have already been generated at step $n-1$. The search for leaf predicates is particularly complex due to the large number of relationships that could hold between the objects in the descriptions. The identification of such predicates is done by representing the observation with a tree (see Figure 1 for an example) in which each level is determined by the propagation of the variables/constants (no relation has to be imposed between two or more predicates at the same level even if they share some variable/constant): the root is the head of the observation and its direct descendants are all the predicates that share with it at least one argument. This procedure is iterated until all the predicates in the description have been inserted in the tree (a considered predicate does not participate anymore to the tree construction). Note that this procedure allows to represent any observation as a tree even when it does not naturally have a tree structure. After the tree is built, the leaf nodes that have only one parent are selected. Let $L = l_1, l_2, \ldots, l_n$ be the set of such leaf predicates: for each element $l_i \in L$ its parent (say $g(a_1, \ldots, a_m)$) is extracted from the tree, and the following rule is generated:

$$rule(a_1, \ldots, a_m) \leftarrow g(a_1, \ldots, a_m), l_i$$

Finally, for each generated rule $R_i = rule_i \leftarrow l_{i_1}, \ldots, l_{i_n}$, if the body $l_{i_1}, \ldots, l_{i_n}$, appears in some rule R_j then $l_{i_1}, \ldots, l_{i_n}$ is replaced in R_j by the predicate $rule_i$ and R_i is eliminated by the set of rules that are being generated. At the end of this step the evaluation phase of the potential rules to make up the Abstraction Theory is performed again according to the procedure above mentioned.

Associating a score to each subset requires a statistical model able to take into account the significance of the subset for the descriptions, i.e. its frequency in them. Specifically, a good subset should have a great discriminating power, i.e. it should be able to discriminate better than any other subset a description from the others. To this aim we exploit the distribution of the subset in the whole set of observations: an high discriminating power means that the subset is fundamental for the concept description since it helps to distinguish a concept from another, while a low discriminating power is interpreted as a hint that the subset is superfluous for the learning process and thus it could be eliminated from the description of the observations. The statistical model that reflects such considerations is the *Term Frequency - Inverse Document Frequency (TF-IDF)* [12], adapted to our work context facing positive and negative observations as follows. For each subset S_i a vector $V_i = (V_{i1}, V_{i2}, \ldots, V_{iN})$ is created, where N is the number of available observations and V_{ij} is the weight of the i-th subset in the j-th observation, computed as:

$$V_{ij} = FREQ_{ij} * (\lg \frac{N}{IFREQ_i} + 1)$$

The term (lg $\frac{N}{IFREQ_i}$ + 1) represents the inverse of the frequency of S_i in the whole set of observations. The result of this computation will be positive if the j-th observation is positive, negative otherwise, thus the resulting vector will be of the form $V_i = (+, -, +, +, -, +, \ldots)$. This allows to distinguish the significance of the subset according to its presence in the positive and negative observations. Now, for each subset we have the vector of its weights in each observation. To select the best subset the following value is computed for each subset:

$$score(S_i) = \sum_{j=1,\ldots,N} V_{ij}$$

It is worth noting that this score will be around zero if the subset equally occurs in both positive and negative observations, in which case it is considered insignificant and could be exploited as a neglecting rule in the abstraction phase. Conversely, an high absolute value indicates a strong correlation of the subset with the positive or the negative observations. Specifically, highly positive (resp., negative) scores indicate that the subset is very frequent in the positive (resp., negative) observations. In both cases, it is considered significant and hence it could be exploited to build shifting rules for the abstraction phase.

Example 3. Let $h(1) : -p(1,2), p(1,4), p(1,5), c(2,3), f(5,6), d(4), s(6)$ the seed chosen in the set of the observations.

- **Step 1**:
  - *Grouping unary predicates:* $S = \emptyset$, no groups of unary predicates with cardinality strictly greater than 1 can be recognized;
- **Step 2**:
  - *Recognize Leaf Nodes:*
    $F = \{c(2,3),\ d(4),\ s(6)\}$, indeed $c(2,3)$ has only one parent $p(1,2)$; $d(4)$ has only one parent $p(1,4)$; $s(6)$ has only one parent $f(5,6)$.
  - *Create the rules - $rule_{l_i}(a_i, ...a_n) \leftarrow g_i, l_i$:*
    $c(2,3)$ with parent $p(1,2) \rightarrow rule1(X,Y) : -p(X,Y), c(Y,Z)$.
    $d(4)$ with parent $p(1,4) \rightarrow rule2(X,Y) : -p(X,Y), d(Y)$.
    $s(6)$ with parent $f(5,6) \rightarrow rule3(X,Y) : -f(X,Y), s(Y)$.
  - *Replace the rule in the set of the observations, for example:*
    $h(1) : -p(1,2), p(1,4), p(1,5), c(2,3), f(5,6), d(4), s(6). \rightarrow$
    $h(1) : -rule1(1,2), rule2(1,4), p(1,5), rule3(5,6)$.
- **Step 3**:
  - *Recognize Leaf Nodes:*
    $F = \{rule3(5,6)\}$, indeed $rule3(5,6)$ has only one parent $p(1,5)$.
  - *Create the rules - $rule_{l_i}(a_i, ...a_n) \leftarrow g_i, l_i$:*
    $rule3(5,6)$ with parent $p(1,5) \rightarrow rule4(X,Y) : -p(X,Y), rule3(Y,Z)$.
  - *Replace the rule in the set of the observations:*
    $h(1) : -rule1(1,2), rule2(1,4), p(1,5), rule3(5,6). \rightarrow$
    $h(1) : -rule1(1,2), rule2(1,4), rule4(5,6)$.
- **Step 4**: END - No more Leaf Nodes can be recognized

Figure 1 reports steps 2 and 3 of the tree and rule construction. The procedure continues with the evaluation step of the generated rules, that are:
$rule1(X,Y) : -p(X,Y), c(Y,Z)$. $rule2(X,Y) : -p(X,Y), d(Y)$.
$rule3(X,Y) : -f(X,Y), s(Y)$. $rule4(X,Y) : -p(X,Y), rule3(Y,Z)$.

Now, suppose that P, the percentage empirically computed on the domain at handle, is equal to 95% and that the Score Percentage of each rule is: $score(1) = 95\%$; $score(2) = 99\%$; $score(3) = 75\%$; $score(4) = 86\%$. Then, $rule1$ and $rule2$ will be shifting rules while $rule3$ and $rule4$ neglecting rules:

$rule1(X,Y) : -p(X,Y), c(Y,Z).$ $\quad$ $rule2(X,Y) : -p(X,Y), d(Y).$
$: -f(X,Y), s(Y).$ $\quad$ $: -p(X,Y), rule3(Y,Z).$

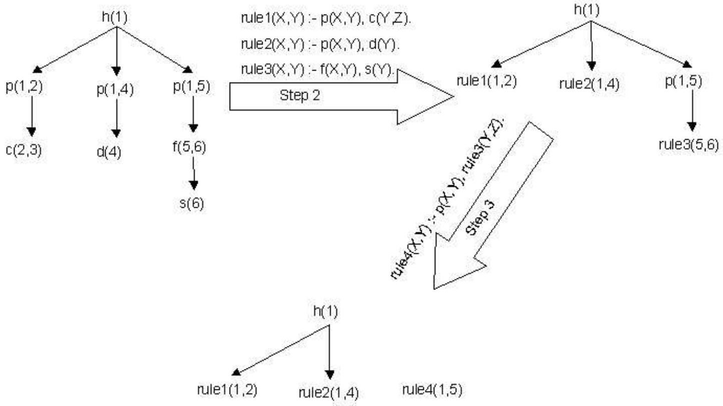


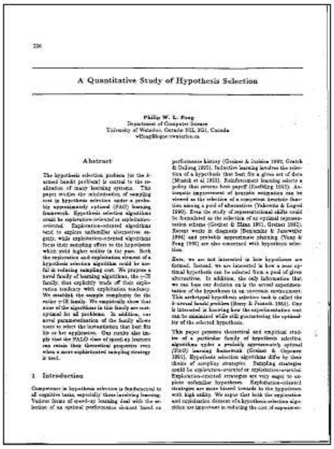
Fig. 1. Tree construction of an observation

4 Experimental Results

The proposed method was implemented in `SICStus Prolog` and tested providing the resulting abstraction theories to the incremental ILP system INTHELEX [4] allowing it to exploit its abstraction capabilities. Various experiments were carried out on a real world application domain of scientific paper documents [5].

The learning tasks to which the learning system was applied, involved the induction of classification rules for 3 classes of scientific papers (96 documents of which 28 formatted according to the International Conference on Machine Learning proceedings (`ICML`), 32 according to the Springer-Verlag Lecture Notes style (`SVLN`) and 36 formatted according to the `IEEET` style), and of rules for identifying the logical components *Author* [36+, 332-], *Page Number* [27+, 341-] and *Title* [28+, 340-] in `ICML` papers (square brackets report the number of positive and negative instances for each label). Figure 2 shows an example of document and its *simplified* description in first order language. 33 repetitions of each learning task were carried out, in each of which the dataset was randomly split into a training set (including 70% of the observations), exploited also to induce the rules for the abstraction operators, and a test set (made up of the remaining 30%).

To build neglecting rules, the threshold for considering low discriminating power (i.e. the score near to zero) was empirically set to ±5% of the minimum positive value and of the maximum of the negative ones in the vector associated



206

A Quantitative Study of Hypothesis Selection

Philip W. L. Fong
Department of Computer Science
University of Waterloo, Ontario N2L 3G1, Canada
wlfong@logos.uwaterloo.ca

Abstract

```
class_icml(icml_1) :-
    part_of(icml_1, icml_2),
    part_of(icml_1, icml_3),
    part_of(icml_1, icml_4),
    part_of(icml_1, icml_5),
   width_very_small(icml_2),
    width_very_large(icml_3),
    width_large(icml_4),
    width_very_large(icml_5),
    height_very_very_small(icml_2),
    height_smallest(icml_3),
    height_very_very_small(icml_4),
    height_smallest(icml_5),
    type_of_text(icml_2),
    type_of_hor_line(icml_3),
    type_of_text(icml_4),
    type_of_hor_line(icml_5),
    pos_right(icml_2),
    pos_center(icml_3),
    pos_center(icml_4),
    pos_center(icml_5),
    pos_upper(icml_2),
    pos_upper(icml_3),
    pos_upper(icml_4),
    pos_upper(icml_5),
    on_top(icml_3, icml_4),
    on_top(icml_3, icml_5),
    on_top(icml_4, icml_5),
    alignment_left_col(icml_3, icml_5),
    alignment_right_col(icml_3, icml_5),
    alignment_center_col(icml_3, icml_5),
```

Fig. 2. Sample ICML document and an extract of its whole description

to the rule. To build shifting rules that have an high discriminating power (i.e. very frequent either in positive or in negative observations only) the threshold was empirically set to the score less then 95% of the minimum positive value and of the maximum of the negative ones in the vector associated to the rule for the classification task and less then 75% of the minimum positive value and of the maximum of the negative ones in the vector associated to the rule for the understanding task.

The average results along with the number of refinements and of clauses learned, the predictive accuracy of the learned theories and the runtime (sec), including both the time for the abstraction step and the learning task, are reported in Table 1. According to a paired *t*-test, there is no statistical difference between the results with and without abstraction, except for runtime. Having the same performance (predictive accuracy) and behavior (no. of clauses and refinements) both with and without abstraction means that the proposed technique was actually able to eliminate superfluous details only, leaving all the information that was necessary for the learning task, which was a fundamental requirement for abstraction. Conversely, runtime was dramatically reduced when using abstraction thanks to the shorter descriptions obtained by eliminating the details, which was exactly the objective of using abstraction. Note that the abstraction theory for a domain is learned once at the beginning of the learning process and is reused every time the learning system is applied on the same domain.

Table 1. System performance exploiting the discovered abstraction theories

	ICML		SVLN		IEEET	
	With Abs	No Abs	With Abs	No Abs	With Abs	No Abs
Lgg	5.81	5.54	7.36	8.12	8.03	8.30
Cl	1.21	1.27	2.75	2.69	2.03	2.27
Accuracy	96.93%	96.75%	86.54%	87.36%	90.69%	90.57%
Runtime	2.00	3.16	11.34	19.46	7.64	27.55

	Author		Page Number		Title	
ICML	With Abs	No Abs	With Abs	No Abs	With Abs	No Abs
Lgg	8.9	8.96	8.15	8.12	8.81	9.09
Cl	2.33	2.06	2.39	2.45	2.42	2.54
Accuracy	97.18%	97.12%	97.81%	97.54%	98.12%	97.87%
Runtime	14.44	29.07	34.06	76.22	27.70	51.67

An example of neglecting rule identified with the proposed strategy is:

```
:- type_graphic(A), pos_upper(A).
```

meaning that a graphics being placed in upper position is not discriminant between positive and negative examples. An example of shifting rule learned is:

```
pos_upper_type_text(A) :- type_text(A), pos_upper(A).
```

As expected, exploiting the abstraction operators the system learns shorter clauses. For instance, the theory learned for *author* contains two clauses made up of 18 and 15 literals (against the 19 and 37 without using abstraction):

```
logic_type_author(A) :- height_medium_small(A), pos_upper_type_text(A),
   part_of(B, A), part_of(B, C), height_very_small_type_text(C),
   pos_upper_type_text(C), part_of(B, D), width_very_large(D),
   height_smallest(D), type_hor_line(D), pos_center_pos_upper(D),
   alignment_left_col(D, E), on_top(F, E), part_of(B, E), part_of(B, F),
   part_of(B, G), type_text_width_medium_large(G), pos_left_type_text(G).
logic_type_author(A) :- part_of(B, A), part_of(B, C),
   pos_upper_type_text(A), pos_center_pos_upper(A),
   pos_upper_type_text(C), pos_left_type_text(C),
   height_very_very_small_type_text(C), on_top(C, D),
   part_of(B, D), on_top(E, A), width_very_large(E), height_smallest(E),
   pos_center_pos_upper(E), on_top(F, E), alignment_center_col(F, E).
```

where the presence of several abstract predicates confirms that the automatically generated abstraction theory was able to identify discriminative intermediate concepts.

5 Conclusion and Future Works

The integration of inference strategies supporting pure induction in a relational learning setting, such as *abstraction* to reason at multiple levels, can be very advantageous both in effectiveness and efficiency for the learning process. In inductive learning, the shift to a higher level representation can be performed directly when the abstraction theory is given and usually an expert domain has to built such a theory. This paper presented a technique for automatically inferring the information needed to apply abstraction operators in an inductive learning framework, exploiting the same observations that are input to the inductive algorithm. Application of the proposed technique in a real learning system proved its viability for significantly improving learning time in complex real-world domains. Future work will concern the analysis of heuristics to choose the seed, to improve the generation of abstraction theories and the design of techniques that can provide information for further abstraction operators.

References

[1] S. Ceri, G. Gottlöb, and L. Tanca. *Logic Programming and Databases.* Springer-Verlag, Heidelberg, Germany, 1990.

[2] L. De Raedt. *Interactive Theory Revision - An Inductive Logic Programming Approach.* Academic Press, 1992.

[3] G. Drastah, G. Czako, and S. Raatz. Induction in an abstraction space: A form of constructive induction. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 708–712, 1989.

[4] F. Esposito, S. Ferilli, N. Fanizzi, T.M.A. Basile, and N. Di Mauro. Incremental multistrategy learning for document processing. *Applied Artificial Intelligence: An Internationa Journal*, 17(8/9):859–883, 2003.

[5] S. Ferilli, N. Di Mauro, T.M.A. Basile, and F. Esposito. Incremental induction of rules for document image understanding. In A. Cappelli and F. Turini, editors, *AI*IA 2003*, volume 2829 of *LNCS*, pages 176–188. Springer, 2003.

[6] N. S. Flann and T. G. Dietterich. Selecting appropriate representations for learning from examples. In *AAAI*, pages 460–466, 1986.

[7] A. Giordana, D. Roverso, and L. Saitta. Abstracting concepts with inverse resolution. In *Proceedings of the 8th International Workshop on Machine Learning*, pages 142–146, Evanston, IL, 1991. Morgan Kaufmann.

[8] A. Giordana and L. Saitta. Abstraction: A general framework for learning. In *Working Notes of the Workshop on Automated Generation of Approximations and Abstractions*, pages 245–256, Boston, MA, 1990.

[9] P.C. Kanellakis. Elements of relational database theory. In J. Van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B of *Formal Models and Semantics*, pages 1073–1156. Elsevier Science Publishers, 1990.

[10] S.H. Muggleton and L. De Raedt. Inductive logic programming. *Journal of Logic Programming: Theory and Methods*, 19:629–679, 1994.

[11] C. Rouveirol and J. Puget. Beyond inversion of resolution. In *Proceedings of ICML97*, pages 122–130, Austin, TX, 1990. Morgan Kaufmann.

[12] G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing and Management*, 24(5):513–523, 1988.

[13] P.E. Utgoff. Shift of bias for inductive concept learning. In R.S. Michalski, J.G. Carbonell, and T.M. Mitchell, editors, *Machine Learning: an artificial intelligence approach*, volume II, pages 107–148. Morgan Kaufmann, Los Altos, CA, 1986.

[14] J.-D. Zucker. Semantic abstraction for concept representation and learning. In R. S. Michalski and L. Saitta, editors, *Proceedings of the 4th International Workshop on Multistrategy Learning*, pages 157–164, 1998.

[15] J.-D. Zucker. A grounded theory of abstraction in artificial intelligence. *Philosophical Transactions: Biological Sciences*, 358(1435):1293–1309, 2003.

[16] J.-D. Zucker and J.-G. Ganascia. Representation changes for efficient learning in structural domains. In L. Saitta, editor, *Proceedings of the 13th International Conference on Machine Learning*, pages 543–551. Morgan Kaufmann, 1996.