HAL
open science

Resource-bounded relational reasoning: induction and deduction through stochastic matching

Michèle Sebag, Céline Rouveirol

To cite this version:

Michèle Sebag, Céline Rouveirol. Resource-bounded relational reasoning: induction and deduction through stochastic matching. Machine Learning, 2000, 38 (1-2), pp.41-62. 10.1023/A:1007629922420 . hal-00111312

HAL Id: hal-00111312

https://hal.science/hal-00111312v1

Submitted on 5 Aug 2019

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Resource-bounded Relational Reasoning: Induction and Deduction Through Stochastic Matching*

MICHÈLE SEBAG
LMS, CNRS UMR 7649, Ecole Polytechnique, France

CÉLINE ROUVEIROL
LRI, CNRS UMR 8623, Université Paris-Sud, France

Editors: Floriana Esposito, Ryszard Michalski, & Lorenza Saitta

Abstract. One of the obstacles to widely using first-order logic languages is the fact that relational inference is intractable in the worst case. This paper presents an any-time relational inference algorithm: it proceeds by stochastically sampling the inference search space, after this space has been judiciously restricted using strongly-typed logic-like declarations.

We present a relational learner producing programs geared to stochastic inference, named STILL, to enforce the potentialities of this framework. STILL handles examples described as definite or constrained clauses, and uses sampling-based heuristics again to achieve any-time learning.

Controlling both the construction and the exploitation of logic programs yields robust relational reasoning, where deductive biases are compensated for by inductive biases, and vice versa.

Keywords: first order logic, bounded reasoning, inductive logic programming

1. Introduction

H. Simon distinguishes two ways of dealing with complex real-world situations (Simon, 1982). In the first one, "the description of the real world is radically simplified until reduced to a degree of complication that the decision maker can handle". The second approach is based on *satisficing*: "Satisficing seeks simplification in a somewhat different direction, retaining more of the detail of the real-world situation, but settling for a satisfactory, rather than approximate-best, decision".

This paper focuses on one particular complex real-world situation: designing Artificial Intelligence (AI) applications in relational domains. As we restrict ourselves to classification applications, this situation is that of Machine Learning (Mitchell, 1997): the goal is to design a set of rules which can be efficiently exploited to classify accurately all or most instances of the problem at hand.

First thing is to choose a representation language. First Order Logic (FOL) languages, based on the theory of Logic Programming (LP) (Lloyd, 1987), allow one to concisely and

*The authors were supported by the ESPRIT project ILP2, LTR 20237.

accurately model many relational domains. For instance, in the domain of chemistry, a molecule is most conveniently expressed as a Prolog clause (Clocksin & Mellish, 1981), describing its atoms and the bonds relating these atoms (Srinivasan et al., 1996). Similarly, in the domain of natural language processing, grammars can be conveniently described in terms of Prolog clauses (Cussens, 1997). Inductive Logic Programming (ILP) (Muggleton & De Raedt, 1994) has extensively studied in the last few years theory and applications of learning programs and concepts in various restrictions of FOL.

On the other hand, the more expressive the language, the more computationally expensive reasoning is within this language (Russell & Norvig, 1995). All languages based on the LP framework share the fact that inference is exponential with respect to the length of the available knowledge.[1] For instance, consider a clause stating that a chemical molecule is toxic if it contains a given pattern of atoms: checking whether this pattern occurs in an actual molecule amounts to graph matching, with exponential complexity in the size of the pattern.

In summary, in a truly relational representation language, inference is intractable unless knowledge can be decomposed into small-size clauses. An alternative is to use languages that are more manageable than those based on Logic Programming, such as KL-1 or its descendants (Brachman, 1977). But, these languages offer less expressive power than LP (Borgida, 1996).

In all such approaches, the complexity of knowledge is "reduced to a degree of complication that the decision maker can handle".

This paper follows instead the other branch of the alternative discussed by Simon, and investigates what a "satisficing" framework for relational applications could be, in the line of resource bounded reasoning (Zilberstein, 1996). Our approach preserves a truly relational description of the domain. But standard inference is replaced by a *satisficing* inference, based on two components: First, expert-supplied declarations are used to better delineate the search space explored by standard inference. These declarations, meant to simplify the structural-matching task (Kodratoff & Ganascia, 1986) are similar in spirit to strongly typed declarations, as used in Escher (Lloyd, 1999), and they similarly restrict the exploration to a meaningful subspace. Still, this subspace remains of exponential size.

The second component of satisficing inference thus is a *stochastic mechanism*, extracting random samples from the inference search space. Whereas standard inference exhaustively explores its search space, satisficing inference only considers a limited number of samples in this space, thus obtaining an incomplete inference procedure with linear complexity in the (user-supplied) number of samples considered. And satisficing inference goes to standard inference when the allotted number of samples goes to infinity.

What remains is to ensure that designing programs/eliciting knowledge fitting this satisficing relational framework can be done in reasonable time. Indeed, the knowledge acquisition was acknowledged to be the bottleneck of AI (Feigenbaum, 1977). This makes it desirable to provide, besides any novel programming framework, some ways of automatically generating programs within this framework.

For these reasons, we describe a relational learner named STILL, first presented in (Sebag & Rouveirol, 1997), that pertains both to ILP and bounded reasoning. STILL constructs constrained logic programs from relational examples (Horn or constrained clauses).

Contrasting with all other ILP learners, STILL is not limited to short clauses: it employs satisficing inductive and deductive inference to extract and exploit (long) clauses within reasonable time and space resources. This way, the syntax biases, (Muggleton, 1995; Giordana & Neri, 1994; Zucker & Ganascia, 1996; Anglano et al., 1998) and search biases (Quinlan, 1990; Pazzani & Kibler, 1992; Muggleton, 1995; Blockeel & Raedt, 1999) traditionally used in ILP to enforce the discovery of short clauses and the learning tractability, are replaced by a stochastic bias.

As far as we know, the works related to bounded reasoning concern either the approximation of an existing knowledge base (Selman & Kautz, 1996; Boufkhad, 1998; Cadoli, 1993) or the restriction of inference (Patel-Schneider, 1990; Crawford & Kuiper, 1989; Crawford & Etherington, 1998). The originality of STILL in this respect is to address both the declarative and the procedural aspects of a relational knowledge-based system. In that sense, STILL fully pertains to the *Learning to Reason* paradigm (Khardon & Roth, 1997): knowledge is captured in whichever format makes its characterization and/or processing more efficient. The limitation of this approach is a loss of intelligibility, due to the fact that clauses are expressed in a compact way for the sake of efficiency. Future work will attempt to remedy this drawback, and determine what further functions should be added to STILL in order for it to display a satisficing intelligibility.

The paper is organized as follows. Section 2 briefly describes the limitations of standard inference. Section 3 shows how type declarations allow one to restrict the search space of inference, and section 4 studies a satisficing inference mechanism exploring this search space. Section 5 is devoted to constructing logic programs geared toward a satisficing inference. Section 6 gives a proof of the principle of the approach potentialities. The paper last situates this work with respect to the state of the art, and sketches some perspectives for further research.

2. Standard relational inference

The core of machine reasoning and learning is the *subsumption procedure* which basically decides whether the premises of a rule (a candidate hypothesis) covers an instance. This section briefly describes and discusses subsumption in a relational language.

2.1. Standard subsumption

The rules and hypotheses considered through the paper are described in a Datalog language, i.e. a definite clause language without function symbols other than constants.[2] Instances are existentially quantified conjunctions of literals as in (de Raedt, Idestam-Almquist, & Sablon, 1997).

For the sake of simplicity, let us consider only the elementary case: checking whether a given clause covers a given instance. In the general case in such languages, the clause is said to *cover* or subsume the instance, if this instance logically implies the body of the clause. A further simplification, consistently used throughout learning in relational logic (Muggleton & De Raedt, 1994), is to use a weaker form of logical implication, namely the θ-subsumption defined by Plotkin (Plotkin, 1970).

Let us recall the definition of θ-subsumption and discuss its limitations, trying to keep the presentation as intuitive as possible. The reader is referred to (Lloyd, 1987) and (Muggleton & De Raedt, 1994) for a comprehensive presentation of LP and ILP.

Definition 1. A formula C θ-subsumes a formula D, noted $D \prec C$, iff there exists a substitution (or *mapping*) σ on variables in C such that $C\sigma$ is included in D ($C\sigma \subseteq D$). C is said to subsume D according to σ.

The mutagenesis domain (King, Srinivasan, & Sternberg, 1995) will be used to illustrate our purpose through the paper (see Section 6). Instances represent molecules, described by a set (a conjunction) of *atom* and *bond* literals.

Example 1: The body of a clause (C) *and an instance description* (D).

$$\begin{array}{ll} C: & atom(Y', carbon, Z'),\ atom(Y'', hydrogen, Z''),\ bond(Y', Y'', simple) \\ D: & atom(d_1, hydrogen, .144),\ atom(d_2, carbon, .014), \\ & atom(d_3, carbon, .33),\ bond(d_2, d_1, simple) \end{array}$$

C *subsumes* D according to $\sigma = \{\, Y'/d_2,\ Z'/.014,\ Y''/d_1, Z''/.144\}$

Subsumption testing is known to be NP-complete (Kapur & Narendran, 1986). In practice, subsumption testing is either affordable or impractical depending on whether the domain description is strongly or weakly structured. Typically, a weakly structured domain is such that instances are described in terms of a small number of predicate symbols, each predicate symbol occurring very often in every instance. Consider for instance the chemistry domain again, and let C be a clause involving 10 *atom* literals and 5 *bond* literals. Let D describe a medium-size molecule, involving 100 atoms and 180 bonds. Checking whether C covers D is in the worst case in $100^{10} \times 180^5 \approx 2 \cdot 10^{31}$.

In strongly structured domains, high-level concepts are used to organize and summarize the information (e.g. the *benzenic-ring* predicate summarizes six carbon atoms and their links). Predicate symbols for high level concepts occur much less often in instances description and as a result, the subsumption complexity dramatically decreases.

2.2. *Limitations of (inductive) Logic Programming*

The preceding section implies that the use of Logic Programming in weakly structured domains is severely limited: only short clauses can be used.

Human programmers prefer writing short clauses as these are easier to debug and more efficient to execute. Learners *cannot* produce long clauses, as it would be intractable to assess them w.r.t. the training set. Practically, ILP learners restrict themselves to only exploring short clauses, by means of syntactic or search biases (see (Nédellec et al., 1996) for a comprehensive presentation of learning biases in ILP):

- Syntax biases: PROGOL (Muggleton, 1995) sets an upper bound on the number of literals in the clauses it produces. REGAL and G-NET (Giordana & Neri, 1994; Anglano et al.,

1998) consider a user-supplied template of the sought hypotheses (fully variabilized clause), and use Genetic Algorithms to optimize domain constraints on the variables. REMO (Zucker & Ganascia, 1996) similarly uses a relational template, termed *morion*, to limit the form of the sought clauses.

- Search biases: FOIL (Quinlan, 1990) and TILDE (Blockeel & Raedt, 1999) follow a top-down approach, and gradually specialize the current hypothesis until its quality is judged satisfactory. The specialization greedily adds the most discriminant literal/variable linking or grounding in each step. The drawbacks of this greedy search are partly overcome by lookahead in FOCL (Pazzani & Kibler, 1992). PROGOL uses a MDL criterion (Rissanen, 1978) favoring short clauses, to stop the search.

To illustrate the practical limitations of induction within FOL, all the above learners restrict themselves to clauses involving up to two or three *atom* literals e.g. in the mutagenesis domain, while example molecules involve up to 40 *atoms* (King, Srinivasan, & Sternberg, 1995).

In such domains, the alternative is to either accommodate short bits of knowledge only, or to design another relational framework.

3. The search space of inference

This section presents some assumptions based on the semantics of relational domains, to restrict the search space explored by standard inference.

3.1. What is relational?

The remainder of the paper relies on the following remark: the arguments of predicates can be divided into two categories depending on the nature of their instantiation domain. Some domains could be modified without in any way affecting the information contained in the examples; for instance, the first argument of predicate *atom*, and the first and second arguments of predicate *bond* have $\{d_1, d_2, d_3, \ldots\}$ for instantiation domain (Example 1). These constants stand for the atom identifiers in the molecules. These constants do not have any meaning in the domain, they are indeed Skolem constants: the semantics of a molecule is invariant up to a consistent renaming of its atoms. Arguments instantiated in such domains are called *relational*.

Other arguments are called *valued*: e.g. the second argument of predicate *atom* indicates the element of the atom and has {*carbon, hydrogen, oxygen*, ...} for instantiation domain. Clearly such domains are not invariant by renaming (permuting *carbon* and *hydrogen* atoms in a molecule would definitely change the nature of the molecule).

We assume the category of the predicate arguments to be explicitly declared by the expert (e.g. *category* (*atom* (*Relational, Valued, Valued*)); *category* (*bond* (*Relational, Relational, Valued*))). These declarations are similar to the *mode* declarations used in PROGOL (Muggleton, 1995) to restrict the learning search to correct (executable) clauses. In contrast, the *category* declarations will be used to restrict the inference search space, i.e.

the set of possible substitutions between a clause and an instance, to correct (meaningful) mappings.

Two more assumptions are made.

- If a predicate does not involve any relational argument, there exists at most one literal built on this predicate in each clause (e.g. the description of a molecule involves at most one literal built on the *lowest-unoccupied-molecular-orbital* predicate). In other words, this predicate corresponds to an attribute.
- If a predicate involves at least one relational argument, there cannot be two literals built on this predicate in a clause, having the same instantiation of the relational arguments, and different instantiations of the valued arguments (e.g. one does not have *atom* (d_1, *carbon*) and *atom* (d_1, *oxygen*)). In other words, the valued arguments in a predicate are functions of the relational arguments of this predicate.

3.2. Functional logic

These assumptions together imply that a relational formula C can be expressed without loss of information as a set of objects and an attribute-value description of some tuples of these objects. For instance, in Example 1 (Section 2.1), and according to the category declarations in the previous section, formula C is described as a set of two objects (the Skolem constants associated to variables Y' and Y'', noted c_1 and c_2) and the value of unary or binary functions of these objects (the atomic type of the objects, e.g. $type(c_1) = carbon$, and the bonds of the objects, e.g. $bond(c_1, c_2) = simple$). Formula D is transformed, or *functionalized*, in the same way:

Example 2: Functional Description of C and D.

$$\boxed{\begin{array}{ll} C\colon \mathcal{O}(C) = \{c_1, c_2\} & [type(c_1) = carbon], \\ & [type(c_2) = hydr.],\ [bond(c_1, c_2) = simple] \\ D\colon \mathcal{O}(D) = \{d_1,\ d_2,\ d_3\} & [type(d_1) = hydr.],\ [charge(d_1) = .144] \\ & [type(d_2) = carbon],\ [charge(d_2) = .014] \\ & [type(d_3) = carbon],\ [charge(d_3) = .33],\ [bond(d_2, d_1) = simple] \\ & [eq_{type,type}(d_2, d_3) = true] \end{array}}$$

Functionalization aims at expressing any given formula as an attribute-value description of object tuples.

Functionalization algorithm. *Let C be a relational formula and C_S its Skolemized form (derived from C by replacing all occurrences of every variable by a Skolem constant). Let $\mathcal{O}(C)$ denote the set of all instantiations of relational arguments in C_S (including Skolem constants), called objects of C. With no loss of generality, each literal t built on a predicate symbol p in C can be expressed as*

$$p\left(o_1^t, \ldots, o_R^t, v_1^t, \ldots, v_V^t\right)$$

where R and V respectively denote the number of relational and valued arguments in p; o_i^t is an object of C; v_j^t instantiates the jth valued argument in p. From Section 3.1, v_j^t can be considered a function of the tuple $o_1^t, \ldots, o_R^t$. Without loss of generality, we assume that p involves exactly one valued variable: e.g. atom(d_1 hydrogen, .144) is rewritten type(d_1 hydrogen), charge(d_1, .144). In case p involves no valued variable, $p(o_1^t, \ldots, o_R^t)$ is rewritten $p(o_1^t, \ldots, o_R^t, true)$.

The relational formula C is finally rewritten into an attribute-value description of object tuples:

- *To each literal $p(o_1^t, \ldots, o_R^t, v^t)$ in C, we associate the attribute $p^\star(o_1^t, \ldots, o_R^t)$, taking value v^t in C. The information conveyed by the variables grounding in C is thus equivalent to the attribute-value conjunction of the $[p^\star(o_1^t, \ldots, o_R^t) = v^t]$.*
- *The links (equality) between relational arguments are directly encoded through the set of objects of C, since objects are in one-to-one correspondence with the instantiations of relational variables.*
- *The links (for both) valued arguments are accounted for by additional attributes. To each pair of valued arguments in C with same instantiation (e.g. $p(o_1^t, \ldots, o_R^t, v^t)$, $p'(o_1^{t'}, \ldots, o_{R'}^{t'}, v^{t'})$ such that $v^t = v^{t'}$), we associate an extra boolean attribute (noted $eq_{p,p'}(o_1^t, \ldots, o_R^t, o_1^{t'}, \ldots, o_{R'}^{t'})$). The functional description of C is thus augmented by the conjunction of $[eq_{p,p'}(o_1^t, \ldots, o_R^t, o_1^{t'}, \ldots, o_{R'}^{t'}) = true]$ over all such attributes.*

Functional logic is defined as follows:

Definition 2. A formula C in functional logic is described as:

- a set of objects $\mathcal{O}(C) = \{o_1, \ldots, o_K\}$,
- a set of constraints of the type: $[g(o_{i_1}, \ldots, o_{i_L}) = V]$, where g is a function of arity L, $(o_{i_1}, \ldots, o_{i_L})$ is a L-tuple of objects in $\mathcal{O}(C)$, and V is a subset of the domain of value of g.

The functionalization algorithm demonstrates that Datalog clauses can be expressed in functional logic with no loss of information. Further, it is clear that functional logic may accommodate various types of links between valued arguments (e.g. domain constraints $[dom(X) = (a, b)]$ or binary constraints $[X < Y]$), making it easy to handle numerical information.

3.3. Discussion

Functionalization can be viewed as being opposite to another change of representation known as *flattening* (Rouveiro1, 1994). Flattening removes all function symbols present in a relational formula, by transforming them into predicate symbols:

$$color(X, blue) \quad \text{becomes} \quad color(X, Y), blue_p(Y)$$

to avoid specific problems related to dealing with function symbols (De Raedt & Džeroski, 1994).

Functional Logic resembles the strongly typed logic language Escher (Lloyd, 1999) in many respects. The difference is the following:

Escher uses type declarations. An elementary type is defined by its value domain (e.g., *atom-label* is valued in $\{d_1, d_2, \ldots\}$; *atom-type* is valued in {*hydrogen*, *oxygen*, ...}). Types are defined by combining previously defined types (e.g. *atom* is composed of *atom-label* and *atom-type*) or as functions between types (e.g. *bond* is a function from a list of atoms onto *bond-type*, which is itself valued in {*simple*, *double*}). Arbitrarily complex terms can be defined.

Functional Logic involves one elementary type (the objects), and only considers functions of object tuples. It does not need any further type declaration, the types being carried by the functions themselves. Its expressiveness is less than that of Escher, though sufficient for all domains considered by ILP so far: chemistry (Srinivasan, 1997; Emde & Wettscherek, 1996) (the objects being the labels of the atoms in a molecule), finite element methods (Dolsak & Muggleton, 1991) (the objects being the labels of the vertices in a mesh), train problems (Michalski, 1983; Botta, Giordana, & Piola, 1997) (the objects being the labels of the cars in a train), family problems (the objects being the names of the persons in a family), etc.

3.4. Subsumption in functional logic

Functionalization is meant to properly delineate the scope of relational vs attribute-value description: what is relational is the fact that the description of formula C is invariant by permutation of $\mathcal{O}(C)$; setting an order on $\mathcal{O}(C)$ induces an attribute-value description of C. Checking whether C subsumes D thus involves two separate elementary steps: considering one mapping σ from $\mathcal{O}(C)$ onto $\mathcal{O}(D)$; comparing the attribute-value description of σC to that of D.

Definition 3. Let C and D be two formulas in functional logic. In the following, $\Sigma(C, D)$, or Σ for short, will denote the set of mappings from $\mathcal{O}(C)$ onto $\mathcal{O}(D)$. A mapping σ of Σ is consistent, if for each constraint $[g(c_{i_1}, \ldots, c_{i_L}) = V]$ in C, there exists a constraint $[g(d_{j_1}, \ldots, d_{j_L}) = W]$ in D, with:

$$\forall k = 1 \ldots L, \sigma\left(c_{i_k}\right) = d_{j_k} \quad \text{and} \quad W \subseteq V$$

Proposition 1. *Let C and D existentially qualified conjunction of literals, and let C and D be the corresponding formulas in functional logic. Then, C θ-subsumes D iff there exists a consistent mapping in $\Sigma(C, D)$.*

The proof straightforwardly follows from the definition of θ-subsumption and the functionalization procedure.

The complexity of subsumption testing in functional logic thus is $\mathcal{O}(n_C \times n_D \times |\Sigma|)$ where: n_C and n_D respectively denote the number of constraints in C and D; $|\Sigma|$ is the number of mappings from $\mathcal{O}(C)$ onto[3] $\mathcal{O}(D)$, hence exponential in the size of $\mathcal{O}(C)$.

4. Satisficing inference

In functional logic, the inference search space is clearly defined though its size remains exponential. An alternative to standard subsumption, first investigated in (Sebag & Rouveiro1, 1997), is to replace the exhaustive exploration of the search space Σ by a sampling-based, Monte-Carlo like, exploration. This section studies the properties and limitations of the satisficing inference.

Practically, one only checks the consistency of some mappings sampled in Σ with uniform probability. This defines a randomized estimate of the subsumption test, called **K-subsumption**.

Definition 4. Let C and D be two formulas in functional logic, and let K be an integer. To each K-tuple $\sigma_1, \ldots, \sigma_K$ in Σ^K, we associate the boolean relation

$$D \prec_{\sigma_1,\ldots,\sigma_K} C$$

taking value *true* iff at least one mapping among the σ_i is consistent. K-subsumption, noted $\prec_K$, is then defined as a stochastic boolean relation, with:

$$\mathbb{P}(D \prec_K C) = \mathbb{P}\big(D \prec_{\sigma_1,\ldots,\sigma_K} C\big)$$

where K-tuples $\sigma_1, \ldots, \sigma_K$ are drawn in Σ^K according to distribution $\mathbb{P}$. Unless specified otherwise, $\mathbb{P}$ will denote the uniform sampling in Σ^K.

Realizing K-subsumption is done with complexity $\mathcal{O}(K \times n_C \times n_D)$. As desired, inference based on K-subsumption, called *satisficing inference*, allows the user to control the time resources of inference through parameter K. And by construction K-subsumption goes to θ-subsumption as K goes to infinity.

Let us examine the approximation cost, that is, the probability that K-subsumption differs from standard θ-subsumption. K-subsumption is obviously correct: if C happens to K-subsumes D, then C θ-subsumes D. As the converse is not true, K-subsumption is not complete.

The chance of error is the probability of having $D \not\prec_K C$ conditioned by $D \prec C$. Let p denote the fraction of mappings in Σ consistent with D; p is strictly positive as $D \prec C$. With regards to the uniform sampling with replacement in Σ, the probability of selecting K mappings *not* consistent is $(1-p)^K$; therefore

$$\mathbb{P}(D \not\prec_K C \mid D \prec C) = (1-p)^K$$

The probability of error thus goes to 0 as K goes to infinity. But unfortunately, the convergence is slow: in our toy example, $\Sigma(C, D)$ contains 9 mappings (from $\{c_1, c_2\}$ onto $\{d_1, d_2, d_3\}$), with $p = 1/9$. Hence, to detect that $D \prec_K C$ with confidence 95%, K must be greater than $\log(.05)/\log(8/9) \approx 26\ldots$

In order to enforce the potentialities of the approach, we thus supply a way of constructing logic programs geared to satisficing inference.

5. Satisficing inductive inference

This section describes the relational learner STILL, first presented in (Sebag & Rouveiro1, 1997). STILL upgrades a propositional learner inspired from the Version Space (Mitchell, 1982), termed *Disjunctive Version Space* (DIVS) (Sebag, 1996), which is first briefly recalled in order for the paper to be self-contained.

5.1. Disjunctive Version Space

We assume the reader's familiarity with the Version Space framework (Mitchell, 1982). This framework has inspired a number of theoretical works (see (Smith & Rosembloom, 1990; Norton & Hirsh, 1993; Smirnov & Braspenning, 1998) among many others), but is severely limited due to its complexity (Haussler, 1988); further, its strict completeness and correction requirements are ill-suited to real problems.

Disjunctive Version Space remedies these limitations by combining the "primitives" $D(E, F)$ of Version Space, where $D(E, F)$ denotes the disjunction (or set) of all hypotheses covering example E and discriminating example F. Let the hypothesis language be constructed from selectors $[att = value]$ (resp. $[att \in interval]$) where *att* denotes a nominal (resp. ordered) attribute. $D(E, F)$ is characterized with linear complexity in the number of attributes, from its upper bound (the disjunction, or set, of all maximally discriminant selectors):

Example 3: Attribute-value examples E and F, and $D(E, F)$

	type	electric-charge	prim-group	sec-group
E	*carbon*	.144	*benzen*	—
F	*oxygen*	.33	—	*methyl*

$$D(E, F) = [type = carbon] \vee [charge < .33] \vee [\text{prim-group} = benzen]$$

The disjunction $VS(E)$ of all correct hypotheses covering any example E might thus be characterized from the conjunction of all $D(E, F_i)$, for F_i ranging over the training examples $F_1, \ldots, F_n$ not belonging to the same target concept as E, termed *counter-examples* of E:

$$VS(E) = D(E, F_1) \wedge \cdots \wedge D(E, F_n)$$

Characterizing $VS(E)$ (or more precisely its upper-bound) from the $D(E, F_i)$, i.e. in Disjunctive Normal Form (DNF), can be done with linear complexity in the number of attributes and the number of examples—while the complexity of $VS(E)$ in Conjunctive Normal Form (CNF) is known to be exponential (Haussler, 1988).

The Disjunctive Version Space is composed of all $VS(E)$ for E ranging over the training set. Further instances U of the problem domain are classified according to the majority vote of the training examples E such that U is covered by, or belongs to, $VS(E)$. The classification procedure is parameterized to accommodate the noise and sparseness of the data (see (Sebag, 1996) for more details):

- Noise. U belongs to $VS(E)$ if it satisfies most (instead of, all) $D(E, F_i)$;
- Sparseness. U satisfies $D(E, F)$ if it satisfies at least M (instead of, one) selectors in $D(E, F)$.

5.2. *Relational Disjunctive Version Space*

Implementing the Disjunctive Version Space in any representation language, only requires the construction of the set $D(E, F)$ of hypotheses covering E and discriminating F. The first thing to be done here is to define discrimination in functional logic.

Definition 5. Let E and F be two formulas in functional form. A mapping σ belonging to $\Sigma(E, F)$ is inconsistent if there exists a pair of constraints $[g(e_{i_1}, \ldots, e_{i_L}) = V]$ in E and $[g(f_{j_1}, \ldots, f_{j_L}) = W]$ in F, such that:

$$\forall l = 1 \ldots L, \sigma\left(e_{i_l}\right) = f_{j_l} \quad \text{and} \quad W \cap V = \phi$$

Definition 6. Let E and F be two formulas in functional form. E discriminates F if all mappings in $\Sigma(E, F)$ are inconsistent.

Let E and F be examples expressed in functional logic (this corresponds to the learning from entailment setting (de Raedt, 1996)).

$$E : \mathcal{O}(E) = \{e_1, e_2, \ldots\}; \qquad \left[g_i\left(e_{i_1}, \ldots, e_{i_K}\right) = U_i\right], \quad i = 1, \ldots, n_E$$
$$F : \mathcal{O}(F) = \{f_1, f_2, \ldots\}; \qquad \left[g'_j\left(f_{j_1}, \ldots, f_{j_L}\right) = V_j\right], \quad j = 1, \ldots, n_F$$

Let $AVL(E)$ denote the attribute-value language defined by attributes $g_i(e_{i_1}, \ldots, e_{i_{K_i}})$, noted g_i for short. In $AVL(E)$, E is simply the conjunction of the selectors $[g_i = U_i]$.

To each σ in $\Sigma(E, F)$ we associate an attribute-value example noted σF, described in $AVL(E)$ as the conjunction of all $[g_i = V_i]$ such that: constraint $[g_i(f_{i_1}, \ldots, f_{i_{K_i}}) = V_i]$ belongs to F; and $\sigma(e_{i_j}) = f_{i_j}$ for $j = 1, \ldots, L$.

Example 4: Functional examples E and F, and language $AVL(E)$

1• Description of E and F.

$E : \mathcal{O}(E) = \{e_1, e_2\}$ $\quad$ $[type(e_1) = carb.]$, $[type(e_2) = hydr]$, $[bond(e_1, e_2) = true]$

$F \quad \mathcal{O}(F) = \{f_1, f_2, f_3\}$ $\quad$ $[type(f_1) = hydr.]$, $[type(f_2) = carb]$, $[type(f_3) = cl]$, $[bond(f_3, f_1) = true]$

2• Expressing $E, \sigma_i F$ and $D(E, \sigma_i F)$ in $AVL(E)$.

for $\sigma_1 = \{e_1/f_3, e_2/f_1\}$ and $\sigma_2 = \{e_1/f_2, e_2/f_1\}$

	$atm(e_1)$	$atm(e_2)$	$bond(e_1, e_2)$
E	*carb.*	*hydr.*	*true*
$\sigma_1 F$	*cl.*	*hydr.*	*true*
$\sigma_2 F$	*carb.*	*hydr.*	—

$D(E, \sigma_1 F) = [type(e_1) = carb.]$

$D(E, \sigma_2 F) = [bond(e_1, e_2) = true]$

We then define $D(E, F)$ in functional logic with respect to the conjunction of the $D(E, \sigma F)$, expressed in $AVL(E)$:

Proposition 2. *A hypothesis H belongs to $D(E, F)$ iff its representation in the attribute-value language defined from E, belongs to (implies) all attribute-value hypotheses $D(E, \sigma F)$, for σ ranging over $\Sigma(E, F)$. For short, we note:*

$$D(E, F) = \bigwedge_{\sigma \in \Sigma(E,F)} D(E, \sigma F)$$

Proof: Without loss of generality, any hypothesis H that covers E is put as:

$$H : \mathcal{O}(H); \qquad \left[g_i\left(e_{i_1}, \ldots, e_{i_K}\right) = W_i\right], \quad i = 1, \ldots, n_H$$
$$\text{with } \mathcal{O}(H) \subseteq \mathcal{O}(E),\ n_H \leq n_E, \quad \text{and} \quad \forall i = 1, \ldots, n_H, U_i \subseteq W_i$$

In language $AVL(E)$, H simply is the conjunction of $[g_i = W_i]$ for $i = 1$ to n_H.
$\Rightarrow$ Assuming that H belongs to $D(E, F)$, it discriminates F, i.e. for every mapping σ in[4] $\Sigma(H, F)$, there exists a constraint $[g(e_{i_1}, \ldots, e_{i_K}) = W_i]$ in H and a constraint $[g(f_{j_1}, \ldots, f_{j_K}) = V_j]$ in F such that:

$$\forall k = 1, \ldots, K, \sigma\left(e_{i_k}\right) = f_{j_k} \quad \text{and} \quad W_i \cap V_j = \phi$$

The above implies that (the $AVL(E)$ description of) H is discriminated from σF, hence H belongs to $D(E, \sigma F)$. As this holds for all σ in $\Sigma(E, F)$,

$$H \in D(E, F) \Longrightarrow H \in \bigwedge_{\sigma \in \Sigma(E,F)} D(E, \sigma F)$$

$\Leftarrow$ Inversely, let H be a hypothesis covering E and assume that H belongs to $D(E, \sigma F)$. Then, there exists an attribute g_i in $AVL(E)$ discriminating (the attribute-value description of) H from σF, hence σ is inconsistent. As H belongs to all $D(E, \sigma F)$, all mappings in $\Sigma(H, F)$ are inconsistent, i.e. H discriminates F. □

In Example 4, one sees from $D(E, \sigma_1 F)$ that e_1 atomic-type must be carbon; and from $D(E, \sigma_2 F)$, that there must be a bond between e_1 and e_2. Taking their conjunction would lead to *there exists a carbon atom linked to some other atom*, which indeed covers E and rejects F.

The complexity of $D(E, F)$ finally is linear in the number n_E of constraints in E, and linear in the size of $\Sigma(E, F)$.

5.3. Discussion

Mapping relational onto attribute-value examples was first explored in a systematic way in the LINUS system (Lavrač & Džeroski, 1994). Several assumptions are made in LINUS to

ensure that each relational example is mapped onto one attribute-value example (one-to-one mapping). In our framework, these assumptions are equivalent to assuming that formulas can be expressed in functional logic with a single object.

Along the same lines, REMO relies on a user-supplied template, termed *morion*, to map every relational example onto (a set of) attribute-value examples (Zucker & Ganascia, 1996). REMO gets rid of the syntactic restrictions used in LINUS: as a relational example may include several occurrences of a given morion, it is accordingly mapped onto as many[5] attribute-value examples.

Note that the morion-based reformulation unfolds a positive relational example into a set of attribute-value examples, thereby creating a Multiple Instance Problem (Dietterich, Lathrop, & Lozano-Perez, 1997). This makes the learning problem significantly more difficult: one searches hypotheses covering at least one, but not necessarily all, attribute-value examples derived from each multiple positive example.

Inversely, the reformulation presented here operates on a local scale: it only considers a single example E and the counter-examples thereof. As the target attribute-value language depends on E itself, this reformulation does not create any such thing as the Multiple Instance Problem: E is transformed into a single attribute-value example in $AVL(E)$ language.

However, our approach is not directly applicable since it maps any counter-example F onto exponentially many attribute-value examples (the size of $\Sigma(E, F)$ is exponential in the size of $\mathcal{O}(E)$).

5.4. *Induction in STILL*

This limitation is again overcome by a sampling-based exploration of $\Sigma(E, F)$. We first construct a stochastic boolean relation, defining a satisficing equivalent of discrimination.

Definition 7. Let C and D be two relational formulas in functional form. To each σ in $\Sigma(C, D)$, we associate the boolean relation noted $\not\prec_\sigma$, taking value true if all σ is inconsistent.

$$D \not\prec_\sigma C$$

Satisficing discrimination, noted $\not\prec_{\text{sat}}$, is then defined as a stochastic boolean relation, with

$$\mathbb{P}(D \not\prec_{\text{sat}} C) = \mathbb{P}(D \not\prec_\sigma C)$$

where σ is drawn in $\Sigma(C, D)$ according to distribution $\mathbb{P}$.

The STILL (for *STochastic Inductive Logic Learning*) algorithm constructs a set $D_K(E, F)$ of hypotheses covering E and approximatively satisficingly discriminating F.

Proposition 3. *E and F being relational formulas, let $\sigma_1, \ldots, \sigma_K$ be a K-tuple selected in $\Sigma(E, F)^K$ with distribution $\mathbb{P} \otimes K$. Then let $D_K(E, F)$, be characterized as:*

$$D_K(E, F) = D(E, \sigma_1 F) \wedge \cdots \wedge D(E, \sigma_K F)$$

$D_K(E, F)$ covers E by construction and the probability that $D_K(E, F)$ satisficingly discriminates F goes to 1 as K goes to infinity.

The proof follows from Definition 7 and Proposition 2.

By taking the conjunction of the $D_K(E, F)$ over all counter-examples F of E, STILL characterizes a set $VS_K(E)$ of hypotheses covering E and approximatively satisficingly discriminating all F. As $VS_K(E)$ is constructed for all training examples E, the complexity of induction in STILL is $\mathcal{O}(N \times K \times n_{\text{Max}})$, N being the number of training examples and n_{Max} an upper bound on the number of constraints in any example.

The user controls the trade-off between the induction cost and the quality of the hypotheses through the number K of samples considered: the computational cost increases linearly with K, and $VS_K(E)$ goes to $VS(E)$ as K goes to infinity.

Any-time Relational Induction	**STILL**
For each E in the training set,	*initialization*
$VS_0(E) = True$;	
For $k = 1$ to K	*interruptible at any-time*
For each E in the training set,	
Select F not belonging to the same target concept as E	
Randomly select σ in $\Sigma(E, F)$	
Construct $D(E, \sigma F)$	*Section 5.1*
$VS_k(E) = VS_{k-1}(E) \wedge D(E, \sigma F)$	

It is worth noting that in most sampling-based learners, the stochastic mechanism only intervenes during the learning phase. For instance in (Kivinen & Mannila, 1995), candidate integrity constraints are checked against some samples of the training data; and provided the number of samples is large enough, the incorrect integrity constraints will almost surely be discarded.

In contrast, STILL uses a stochastic sampling mechanism during both the learning and the exploitation phases: even though they were constructed in polynomial time, hypotheses in $VS_K(E)$ cannot be exploited by standard subsumption in polynomial time. This is because the hypotheses learned are potentially as long as the examples; therefore, they can only be exploited, e.g. to classify further instances, through satisficing inference (Section 4).

6. Validation: a proof of the principle

It is long known that instances of NP-hard problems are not all equally hard (Mitchell, Selman, & Levesque, 1992): the actual complexity matches the worst-case complexity under some particular circumstances. This phenomenon, known as *Phase Transition*, has been extensively studied in the framework of relational subsumption, on artificial problems (Giordana et al., 1999).

In this section, we study instead how satisficing inference addresses the problem of relational learning and reasoning in an actual domain, well-known as mutagenesis problem (King, Srinivasan, & Sternberg, 1995), which has been used as benchmark for all ILP learners.

6.1. *The domain and reference results*

The mutagenesis problem is concerned with nitroaromatic molecules occurring in car exhaust fumes; some of them have a carcinogenic effect due to their high mutagenicity, and the literature has not yet offered any theory for predicting the mutagenicity of molecules. By the way, the field of chemistry and particularly predictive toxicology evaluation, offers many inspiring and challenging applications for Machine Learning (Srinivasan, 1997).

The following will only consider the subset of 188 molecules known as *regression-friendly*. Several descriptions of the mutagenicity domain have been considered. An attribute-value description of the domain (background knowledge $\mathcal{B}_0$) has been considered by CART (Breiman et al., 1984), linear regression (LR) and 1-hidden layer neural nets (NN). Their predictive accuracy estimated by 10fold cross-validation and reported from (Srinivasan & Muggleton, 1995), ranges from 88 ± 2 percent (CART) to 89 ± 2 percent (LR, NN).

Several relational descriptions of the domain are available. A weakly structured description termed $\mathcal{B}_2$ (Srinivasan & Muggleton, 1995) only involves the atoms and bonds of the molecules:

$$atm(m_1, c, 22, -.116), \ldots, atm(m_2, o, 40, -.386), bond(m_1, m_2, simple), \ldots,$$

A refined description $\mathcal{B}_3$ involves four additional boolean or numerical attributes, e.g. describing the hydrophobicity of the molecule. A strongly structured description $\mathcal{B}_4$ also involves high level chemical concepts (e.g. benzenic rings, nitric groups) that are present in the molecules.

As noted earlier on, rewriting $\mathcal{B}_2$, $\mathcal{B}_3$ or $\mathcal{B}_4$ in functional form is straightforward: the objects in a molecule are its atom names m_i, and all information can be expressed through functions of these objects.

Relational learners display a high degree of sensitivity with respect to the description of the domain (Table 1). Typically, Progol results increase from $81\% \pm 3$ for the weakly structured description $\mathcal{B}_2$, to $88\% \pm 2$ for the strongly structured $\mathcal{B}_4$. Simultaneously, the run time decreases from 64,350 seconds (on HP-735 workstation) with $\mathcal{B}_2$ to 40,950 with $\mathcal{B}_4$.

6.2. *Experimental aim and setting*

Our primary intention is to see whether reasonable results can be obtained within a reasonable time on weakly structured domains, using satisficing inductive and deductive inference. We therefore only consider background knowledges $\mathcal{B}_2$ and $\mathcal{B}_3$.

Table 1. Reference results on the 188 dataset.

	$\mathcal{B}_2$	$\mathcal{B}_3$	$\mathcal{B}_4$
FOIL (Quinlan, 1990)	61 ± 6	83 ± 3	86 ± 3
Progol (Muggleton, 1995)	81 ± 3	83 ± 3	88 ± 2
Fors (Karalic, 1995)	NA	NA	89 ± 6
G-Net (Anglano et al., 1998)	NA	NA	92 ± 8

A second point regards the sampling mechanism used to stochastically explore the inference search space. Satisficing inference is defined with respect to uniform sampling without replacement of $\Sigma(E, F)$: while possible, an atom e_i in E is drawn with uniform probability without replacement in $\mathcal{O}(E)$ and mapped onto some atom f_j in F, selected with uniform probability without replacement in $\mathcal{O}(F)$.

We can also incorporate some knowledge in this sampling mechanism, by selecting f_j depending on e_i: typically, one would rather map a carbon atom in E onto a carbon atom in F. The uniform selection of f_j is thus replaced by a deterministic selection of the not yet selected atom in $\mathcal{O}(F)$ that is "most similar"[6] to e_i.

STILL involves four parameters:

- The number K and K' of samples considered respectively by inductive and deductive stochastic inference (Sections 5 and 4). K and K' are respectively set to 300 and 1; complementary experiments (not shown here) demonstrate that doubling K and K' does not significantly improve the results.
- Parameters ε and M used to control respectively the consistency and the generality of the hypotheses (with $\varepsilon = 0$ for perfect consistency, and $M = 1$ for maximal generality; see (Sebag, 1996) for more details); ε varies in [0, 2]; M varies in [1, 7].

STILL predictive accuracy is evaluated by 10 fold cross-validation. As recommended for stochastic algorithms, the result associated to each fold is averaged over 15 independent runs (same parameter setting, distinct random seeds). STILL is written in C^{++}.

6.3. *Results and discussion*

Table 2 shows the results obtained with uniform sampling on $\mathcal{B}_2$ and $\mathcal{B}_3$, for M ranging in $\{1, 2\}$ and ε ranging in $\{0, 1, 2\}$. Run-times are given in seconds on a Pentium II 166.

The average accuracy is fairly stable from one run to another; the high variance of the results is due to the fact that some folds (always the same) are easy for STILL (e.g. the 6th) and others are difficult (e.g. the first). An automatic adjustment of parameters M and ε (e.g., as proposed in (Kohavi & John, 1995)) is highly desirable, as these parameters command the overall performance of the algorithm. Still, the above experiments demonstrate that satisficing results can be obtained through stochastic relational inference on this problem:

Table 2. STILL results with uniform sampling, $K = 300$, $K' = 1$.

	M	ε	Accuracy
Uniform Sampling on $\mathcal{B}_2$	1	0	73.7 ± 9.3
(time: 32 seconds)	1	1	80.8 ± 7.3
	1	2	83 ± 7.9
Uniform Sampling on $\mathcal{B}_3$	2	0	86.5 ± 6.3
(time: 27 seconds)	2	1	84.2 ± 8.1
	2	2	82.2 ± 9.3

Table 3. STILL results with informed sampling, $K = 300$, $K' = 1$.

	M	ε	Accuracy	Time
Informed sampling on $\mathcal{B}_2$	3	0	85.2 ± 7.6	40
	4	0	86.2 ± 7.7	43
	5	0	86.5 ± 6.9	45
	6	0	85.2 ± 6.3	47
Informed sampling on $\mathcal{B}_3$	3	0	85 ± 8.1	34
	4	0	86.9 ± 7.5	38
	5	0	88.6 ± 8	41
	6	0	88.8 ± 7	43

the computational cost is lower by three orders of magnitude compared to the state-of-the-art ILP learner PROGOL. STILL, processing a poor description of the domain, obtains results comparable to those obtained by PROGOL and FOIL, that were processing a richer description. As might have been expected, the optimal value of M (the right degree of specificity of the hypotheses) increases as more information is available (from $\mathcal{B}_2$ to $\mathcal{B}_3$).

Interestingly, the results obtained with an informed sampling are better, but not much more than with uniform sampling (Table 3).

Obviously, many more experiments need to be done to study the "niche" of satisficing inference. These limited experiments were only meant to show it was worth the study. In particular, further work will examine how ε and M are related to the example noise and distribution, and how their optimal value could be estimated.

7. Related work

The satisficing approach pervades many areas of Artificial Intelligence and Operational Research (Zilberstein, 1996). Satisficing reasoning, also termed bounded reasoning, has been explored along two main lines in logic. In the first line, we call it *off-line* or syntactical, the goal is to put a knowledge base (KB) into a form allowing one to efficiently answer all (or most) queries using standard inference. Pertaining to this field is knowledge compilation (Reiter & Kleer, 1987; Marquis, 1995). Other works are devoted to finding tractable upper and lower bounds of a given intractable KB ($KB_l \prec KB \prec KB_u$). These bounds allow one to answer part of the queries with tractable complexity ($KB \models c$ if $KB_l \models c$ and $KB \not\models c$ if $KB_u \not\models c$) (Selman & Kautz, 1996). The point is to find the best bounds, i.e. the best space of tractable approximations of an intractable KB (Boufkhad, 1998). However, the cost of compilation seems to be taken into minor consideration since compilation is viewed as an off-line process (Cadoli, 1993).

In the second line, we call it *on-line* or operational, the KB is left unchanged, but the standard inference is restricted (and becomes thus incomplete) in such a way that deduction from this KB becomes tractable. The point is to identify the causes of intractability and to prevent their activation. For instance, the length of the deduction chains might

be restricted (Patel-Schneider, 1990); one could restrict the exploration to some paths of the KB (Crawford & Kuiper, 1989), or deductions beyond the bounds of a fixed context could be pruned (Crawford & Etherington, 1998). One drawback is that the time needed by meta-reasoning (e.g. deciding which branches of the reasoning tree can be cut), might offset the time actually gained at the reasoning level.

Our work tackles a different issue. First of all, it concerns predicate logic while bounded reasoning has been mainly restricted to propositional logic so far.

Second, the cited works either concern the reformulation of the KB, or the design of a tractable inference algorithm. However, the overall efficiency of a KBS depends on the KB, on the inference engine, and on how both fit together. As far as efficiency is concerned, the syntactical and the operational aspects should be considered together. The main novelty of the presented work is to address, in the same satisficing spirit, the construction and the exploitation of a relational KB. Reasoning biases become unavoidable during the exploitation of a KB, as standard inference is replaced by satisficing, classically-incomplete inference. And biases are also unavoidable during the construction of a KB, be they machine learning or human expert biases. Taking in charge both the construction and the exploitation of a KBS allows both sources of bias to counteract each other.

Our work closely resembles the Learning to Reason (L2R) framework, as opposed to the more traditional Learning to Classify (L2C) one (Khardon & Roth, 1997). L2C is essentially concerned with producing knowledge in order to fuel an existing reasoning system; this requires using CNF representations; but, as shown in Section 5.1, CNF representations might cause interesting concepts to be intractable. In opposition, Learning to Reason focuses on acquiring reliable and usable knowledge; it does so by using some interface to the world, based on oracles and queries.

The difference between Learning to Reason and Satisficing Inference is twofold. First, L2R takes advantage of some interface to the world, offering sophisticated ways to alter the distribution of the examples depending on the current hypotheses of the system, and guide the hypothesis construction. In opposition, satisficing inference works so far in batch mode only, handling all examples at once.

Second, L2R only considers propositional representations, where learnable concepts are *de facto* usable. In opposition, satisficing inference is concerned with relational representations: one can polynomially learn approximate concepts, which cannot yet be exploited with polynomial complexity (Section 5.4).

8. Perspectives and satisficing intelligibility

The pitfall of our approach is a severe loss of intelligibility at the operational and syntactical levels. At the operational level, satisficing inference is bound to be less intelligible than standard inference, due to its stochasticity.

At the syntactical level, the theory produced by STILL is compactly expressed in DNF form. Our future work will be to remedy this lack of intelligibility. And, as the notion of intelligibility can hardly be captured by absolute criteria, we distinguish three profiles of users. Addressing their demands defines several perspectives for further research. One perspective concerns scientific discovery (Srinivasan et al., 1996): the scientist user waits

for an insight into the nature of the problem. To this aim, a first step would be to reduce the dimensionality of the problem, as in (Sebag, 1994; Lavrač, Gamberger, & Džeroski, 1995). The scientist may find the selection of the most important features to be almost as insightful as the discovery of the rules.

Another perspective is that of the knowledge engineer, who wants to debug an existing KBS. Assuming that the knowledge engineer is assisted by a machine learning system, one way of refining a KB is to provide new examples. But, determining which new examples would be most helpful, can be done without an intelligible theory. Instead, the actual theory can be used to determine the most informative instances to be labelled by the expert. In this perspective, the ideas of theory revision will be adapted to the revision of a dataset.

A third perspective is that of the end-user, who asks the system to explain its classification of the current instance. If the KB were in intelligible form, the answer would be a general rule, covering the instance at hand. But again, the actual theory can be used to compute a most general and informative rule covering the example at hand. We have called weak-intelligibility the aptitude for a system to explain each of its judgments by an intelligible excerpt of its unintelligible theory (Sebag, 1995). It is worth noting that most human experts, though unable to write down their entire knowledge, use weak intelligibility in a satisficing way. . .

Hopefully, the user's demands regarding the intelligibility of knowledge construction and exploitation can be formulated in terms of questions to be answered. (Expressing one's needs in such operational terms might eventually prove easier than setting requirements on the form of the acceptable theory). Answering these questions in a satisficing way would be a sign of satisficing machine intelligence.

Acknowledgments

We are indebted to A. Srinivasan, S. Muggleton and R. King, who constructed the mutagenesis dataset and made it publicly available. We wish to thank John Lloyd for insightful comments on the edge of relational representations. Thanks to Marc Schoenauer, Josselin Garnier, Ecole Polytechnique, and Yves Kodratoff, Fabien Torre, LRI, for their support and comments. Thanks to the anonymous referees, who pointed out very interesting related works, and to Lise Fontaine, who helped out making the paper readable.

Notes

1. Another weakness is that LP poorly handles numerical information. This limitation can be addressed by using a Constraint Logic Programming formalism, which is a superset of LP supporting interpreted predicates (Jaffar & Lassez, 1987).
2. Clauses including domain and binary constraints (e.g. $[X \in (a, b)]$, $[X < Y]$) in order to better handle numerical information, will be considered in 3.2.
3. Among other attempts to restrict the complexity of inference, let us mention *Subsumption under Object Identity* (Esposito et al., 1996) which would correspond to considering only injective mappings in Σ. One sees that this restriction decreases the cost of subsumption, yet without making it tractable.
4. As $\mathcal{O}(H)$ is included in $\mathcal{O}(\mathcal{E})$, $\Sigma(H, F)$ is naturally embedded in $\Sigma(E, F)$. No distinction is made in the following.

5. Assuming that the current morion includes m objects (as defined in 3.1), an example with p objects is mapped onto $p!/p-m!$ attribute-value examples.
6. Such preferences naturally derive from the description of atoms: Choose f_j such that it has the same electric-charge (e.g., $-.116$) as e_i; otherwise, the same atomic-number (e.g., 22); otherwise, the same atomic-type (e.g., carbon).

References

Anglano, C., Giordana, A., Lo Bello, G., & Saitta, L. (1998). An experimental evaluation of coevolutive concept learning. In J. Shavlik (Ed.), *Proc. of the 15th ICML* (pp. 19–27). Morgan Kaufmann.

Blockeel, H. & De Raedt, L. (1998). Top-down induction of first order logical decision trees. *Artificial Intelligence, 101*(1–2), 285–297.

Borgida, A. (1996). On the relative expressiveness of description logics and predicate logics. *Artificial Intelligence, 82*, 353–367.

Botta, M., Giordana, A., & Piola, R. (1997). FONN: Combining first order logic with connectionist learning. In D. Fisher (Ed.), *Proc. of ICML-97* (pp. 48–56). Morgan Kaufmann.

Boufkhad, Y. (1998). Algorithms for propositional KB approximation. *Proc. of AAAI-98* (pp. 280–285). AAAI Press.

Brachman, R. J. (1977). *A structural paradigm for representing knowledge*. Ph.D. Thesis, Harvard University, Cambridge, MA.

Breiman, L., Friedman, J. H., Olshen, R. A., & Stone, C. J. (1984). *Classification and Regression by tree*. Belmont California: Wadsworth.

Cadoli, M. (1993). Semantical and computational aspects of Horn approximations. *Proc. of IJCAI-93* (pp. 39–44). Morgan Kaufmann.

Clocksin, W. F. & Mellish, C. S. (1981). *Programming in Prolog*. New-York: Springer-Verlag.

Crawford, J. M. & Etherington, D. W. (1998). A non-deterministic semantics for tractable inference. *Proc. of AAAI-98* (pp. 286–291). AAAI Press.

Crawford, J. M. & Kuiper, B. (1989). Toward a theory of access-limited logic for knowledge representation. *Proc. of KR-89* (pp. 67–78).

Cussens, J. (1997). Part-of-speech tagging using Progol. In N. Lavrac & Saso Dzeroski (Eds.), *Proc. of ILP-97* (pp. 93–108). LNCS, Vol. 1297, Springer Verlag.

De Raedt, L. (1996). Induction in logic. In *Proc. of 3rd International Workshop on Multistrategy Learning* (pp. 29–38). AAAI Press.

De Raedt, L. & Džeroski, S. (1994). First order *jk*-clausal theories are PAC-learnable. *Artificial Intelligence, 70*, 375–392.

De Raedt, L., Idestam-Almquist, P., & Sablon, G. (1997). Θ-subsumption for structural matching. In M. Van Someren & G. Widmer (Eds.), *Proc. of the 9th European Conference on Machine Learning* (pp. 73–84). Springer Verlag.

Dietterich, T. G., Lathrop, R., & Lozano-Perez, T. (1997). Solving the multiple-instance problem with axis-parallel rectangles. *Artificial Intelligence, 89*(1–2), 31–71.

Dolsak, D. & Muggleton, S. (1991). The application of ILP to finite element mesh design. In S. Muggleton (Ed.), *Proc. of the 1st ILP* (pp. 453–472). Springer Verlag.

Emde, W. & Wettscherek, D. (1996). Relational instance based learning. In L. Saitta (Ed.), *Proc. of the 13th ICML* (pp. 122–130). Morgan Kaufmann.

Esposito, F., Laterza, A., Malerba, D., & Semeraro, G. (1996). Locally finite, proper and complete operators for refining datalog programs. In Z. W. Raz & M. Michalewicz (Eds.), *Foundations of intelligent systems* (pp. 468–478). Springer Verlag.

Feigenbaum, E. (1977). The art of artificial intelligence: themes and case studies in knowledge engineering. *Proc. of IJCAI'77* (pp. 1014–1029).

Giordana, A. & Neri, F. (1994) Search intensive concept induction. *Evolutionary Computation, 3*(4), 375–416.

Giordana, A., Botta, M., & Saitta, L. (1999). An experimental study of phase transitions in matching. In *Proc. of IJCA'99* (pp. 1199–1203). Morgan Kaufmann.

Haussler, D. (1988). Quantifying inductive bias: AI learning algorithms and Valiant's learning framework. *Artificial Intelligence, 36*, 177–221.

Jaffar, J. & Lassez, J. L. (1987). Constraint logic programming. *Proc. of the fourteenth ACM Symposium on the Principles of Programming Languages* (pp. 111–119).

Kapur, D. & Narendran, P. (1986). NP-completeness of the set unification and matching problems. *Proc. of 8th Conference on Automated Deduction* (Vol. 230, pp. 489–495). Springer Verlag.

Karalic, A. (1995). *First order regression*. Ph.D. Thesis, Institut Josef Stefan, Ljubljana, Slovenia.

Khardon, R. & Roth, D. (1997). Learning to reason. *Jal of the ACM, 44*(5), 697–725.

King, R. D., Srinivasan, A., & Sternberg, M. J. E. (1995). Relating chemical activity to structure: an examination of ILP successes. *New Gen. Comput., 13*, 411–433.

Kivinen, J. & Mannila, H. (1995). Approximate inference of functional dependencies from relations. *Theoretical Computer Science, 149*, 129–149.

Kodratoff, Y. & Ganascia, J.-G. (1986). Improving the generalization step in learning. In R. S. Michalski, J. G. Carbonell, & T. M. Mitchell (Eds.), *Machine learning: an artificial intelligence approach* (Vol. 2, pp. 215–244). Morgan Kaufmann.

Kohavi, R. & John, G. H. (1995). Automatic parameter selection by minimizing estimated error. In A. Prieditis, & S. Russell (Eds.), *Proc. of the 12th ICML* (pp. 304–312). Morgan Kaufmann.

Lavrač, N. & Džeroski, S. (1994). *Inductive logic programming: techniques and applications*. Ellis Horwood.

Lavrač, N., Gamberger, D., & Džeroski, S. (1995). An approach to dimensionality reduction in learning from deductive databases. In L. de Raedt (Ed.), *Proc. of Inductive Logic Programming-95* (pp. 337–354). Katholieke Universiteit Leuven.

Lloyd, J. W. (1987). *Foundations of logic programming*. 2nd edn., Springer-Verlag.

Lloyd, J. W. (1999). Programming in an integrated functional and logic language. *Journal of Functional and Logic Programming*, March 1999.

Marquis, P. (1995). Knowledge compilation using prime implicants. *Proc. of IJCAI-95* (pp. 837–843). Morgan Kaufmann.

Michalski, R. S. (1983). A theory and methodology of inductive learning. In R. S. Michalski, J. G. Carbonell, & T. M. Mitchell (Eds.), *Machine learning: an artificial intelligence approach* (Vol. 1, pp. 83–134). Morgan Kaufmann.

Mitchell, T. M. (1982). Generalization as search. *Artificial Intelligence, 18*, 203–226.

Mitchell, T. M. (1997). *Machine learning*. McGraw Hill.

Mitchell, D., Selman, B., & Levesque, H. (1992). Hard and easy distributions of SAT problems. *Proc. of AAAI-92* (pp. 459–465). AAAI Press.

Muggleton, S. (1995). Inverse entailment and PROGOL. *New Gen. Comput., 13*, 245–286.

Muggleton, S. & De Raedt, L. (1994). Inductive logic programming: Theory and methods. *Journal of Logic Programming, 19*, 629–679.

Nédellec, C., Rouveirol, C., Adé, H., Bergadano, F., & Tausend, B. (1996). Declarative bias in ILP. In L. de Raedt (Ed.), *Advances in ILP* (pp. 82–103). IOS Press.

Norton, S. W. & Hirsh, H. (1993). Learning DNF via probabilistic evidence combination. In P. Utgoff (Ed.), *Proc. of the 10th ICML* (pp. 220–227). Morgan Kaufmann.

Patel-Schneider, P. F. (1990). A decidable first-order logic for knowledge representation. *Journal of Automated Reasoning, 6*, 361–388.

Pazzani, M. & Kibler, D. (1992). The role of prior knowledge in inductive learning. *Machine Learning, 9*, 54–97.

Plotkin, G. (1970). A note on inductive generalization. *Machine intelligence* (Vol. 5, pp. 153–163). Edinburgh University Press.

Quinlan, J. R. (1990). Learning logical definition from relations. *Machine Learning, 5*, 239–266.

Reiter, R. & De Kleer, J. (1987). Foundations of assumption-based truth maintenance systems: preliminary report. *Proc. of AAAI-87* (pp. 183–188). AAAI Press.

Rissanen, J. (1978). Modelling by shortest data description. *Automatica, 14*, 465–471.

Rouveirol, C. (1994). Flattening and saturation: Two representation changes for generalisation. *Machine Learning, 14*, 219–232.

Russell, S. & Norwig, A. (1995). *Artificial intelligence, a modern approach*. Prentice Hall.

Sebag, M. (1994). Using constraints to building version spaces. In L. De Raedt & F. Bergadano (Eds.), *Proc. of ECML-94, European conference on machine learning* (pp. 257–271). Springer Verlag.

Sebag, M. (1995). 2nd order understandability of disjunctive version spaces. In *Workshop on machine learning and comprehensibility, IJCAI-95*. Report LRI, Université Paris-Sud, http://www.lri.fr/ cn/wshp-ijcai.

Sebag, M. (1996). Delaying the choice of bias: A disjunctive version space approach. In L. Saitta (Ed.), *Proc. of the 13th ICML* (pp. 444–452). Morgan Kaufmann.

Sebag, M. & Rouveirol, C. (1997). Tractable induction and classification in FOL. *Proc. of IJCAI-97* (pp. 888–892). Morgan Kaufmann.

Selman, B. & Kautz, H. (1996). Knowledge compilation and theory approximation. *Jal of the ACM, 43*(2), 193–224.

Simon, H. (1982). *Models of bounded rationality*. MIT Press: Cambridge.

Smirnov, E. N. & Braspenning, P. J. (1998). Version space learning with instance-based boundary sets. H. Prade (Ed.), *Proc. of European Conf. on Artificial Intelligence*. John Wiley and Sons.

Smith, B. D. & Rosembloom, P. S. (1990). Incremental non-backtracking focusing: A polynomially bounded generalization algorithm for version space. *Proc. of AAAI-90* (pp. 848–853). Morgan Kaufmann.

Srinivasan, A. & Muggleton, S. (1995). Comparing the use of background knowledge by two ILP systems. In L. de Raedt (Ed.), *Proc. of Inductive Logic Programming-95*. Katholieke Universiteit Leuven.

Srinivasan, A., Muggleton, S. H., Sternberg, M. J. E., & King, R. D. (1996). Theories for mutagenicity: a study in first order and feature-based induction. *Artificial Intelligence, 85*, 277–299.

Srinivasan, A. (1997). The predictive toxicology evaluation challenge. *Proc. of IJCAI-97* (pp. 4–8). Morgan Kaufmann.

Zilberstein, S. (1996). Using any-time algorithms in intelligent systems. *AI Magazine, 17*(3), 73–83.

Zucker, J.-D. & Ganascia, J.-G. (1996). Representation changes for efficient learning in structural domains. In L. Saitta (Ed.), *Proc. of the 13th ICML* (pp. 543–551). Morgan Kaufmann.