Ideal Refinement of Datalog Programs
Giovanni Semeraro, Floriana Esposito and Donato Malerba
Dipartimento di Informatica - Università degli Studi di Bari
Via E. Orabona 4 - 70126 Bari, Italy
{semararo, esposito, malerbad}@vm.csata.it

Abstract. In model inference, refinement operators are exploited to change in an automated way incorrect clauses of a logic program. In this paper, we present two refinement operators for Datalog programs and state that both of them meet the properties of local finiteness, properness, and completeness (ideality). Such operators are based on the quasi-ordering induced upon a set of clauses by the generalization model of \( \theta \)-subsumption under object identity. These operators have been implemented in a system for theory revision that proved effective in the area of electronic document classification.

1 Introduction
In a logical framework for the inductive inference of theories from facts, a fundamental problem is the definition of locally finite, proper, and complete (ideal) refinement operators. Indeed, when the aim is to identify in the limit a finite axiomatization of a theory - a model - expressed in a logical language, it becomes relevant to develop incremental operators that allow for the refinement of theories which turn out to be too weak or too strong.

Shapiro [31] presented a framework for the inductive inference of logic theories. He developed an incremental inductive inference algorithm, the Model Inference System (MIS), that can identify in the limit any model in a family of complexity classes of first order theories. Despite of its scientific importance, this system proved very inefficient on real-world tasks and was able to infer only simple logic programs from a small number of examples.

The ideality of the refinement operators plays a key role when the efficiency and the effectiveness of the learning systems is an unnegligible requirement [34].

Theoretical studies in Inductive Logic Programming (ILP) have shown that, when full Horn clause logic is chosen as representation language and either \( \theta \)-subsumption or implication is adopted as generalization model, there exist no ideal refinement operators [34, 35]. Research efforts in the area of ILP have been directed to improve the efficiency of the learning systems by restricting full first order Horn clause logic by means of suitable language biases, such as linkedness [15], \( ij \)-determinacy [21], Datalog (i.e., function-free) clauses [23, 27], rule models [17], antecedent description grammars [3], clause sets [1] and literal templates [33]. However, these language biases are not sufficient to solve the problem of defining ideal refinement operators. Indeed, as Niblett says [22], "it is an open question as to which restrictions on full first order logic are compatible with complete non-redundant refinement operators."

In this paper, we define two ideal refinement operators for the space of Datalog clauses. These definitions rely on a weaker ordering than \( \theta \)-subsumption, namely \( \theta_0 \)-subsumption.

In Section 2, we briefly recall the definition of the generalization model based on \( \theta_0 \)-subsumption and give the basic definitions concerning the refinement operators. The non-existence of ideal refinement operators for Datalog clauses under \( \theta \)-subsumption is
investigated in Section 3. Novel refinement operators for Datalog clauses ordered by \(\theta_{ol}\)-subsumption are defined in Section 4. Moreover, we point out that such operators are ideal. Section 5 presents an application of these operators to the real-world task of electronic document classification.

2 Preliminaries and Definitions

We assume the reader to be familiar with the notions of substitution, literal, fact, Horn clause and definite clause [19]. A clause \(C = l_1 \lor l_2 \lor \ldots \lor l_n\), is considered as the set of its literals, that is, \(C = \{ l_1, l_2, \ldots, l_n \}\). \(|C|\) denotes the number of literals in \(C\) - the length of \(C\) - while \(\text{size}(C)\) denotes the number of symbol occurrences in \(C\) (excluding punctuation) minus the number of distinct variables occurring in \(C\). Furthermore, we will denote with \(\text{vars}(C), \text{consts}(C),\) and \(\text{terms}(C)\) the set of the variables, of the constants and of the terms occurring in \(C\), respectively. Henceforth, any two clauses will be always assumed to be variable disjoint. This does not limit the expressiveness of the adopted language since any two non-variable disjoint clauses always can be standardized apart.

Henceforth, by clause we mean Datalog clause. Datalog is a language for deductive databases. Here, we refer to [16] for what concerns the basic notions about deductive databases. The vocabulary of a Datalog program \(P\) is composed of intensional database symbols, denoted with \(\text{IDB}'s\), and extensional database symbols, denoted with \(\text{EDB}'s\). \(P\) is made up of a set of Datalog clauses of the form

\[Q(x_1, x_2, \ldots, x_n) : \varphi\]

where:

- the head \(Q(x_1, x_2, \ldots, x_n)\) consists of an IDB \(Q\) of arity \(n\), \(n \geq 0\), and of a list of \(n\) arguments
- the body \(\varphi\) is a set of literals, which can be equality atoms and relational atoms. Both these kinds of atoms must be positive in Datalog. Negations of such atoms are allowed in Datalog\(^-\). If the only negations are inequalities in the bodies, we have a sublanguage of Datalog\(^-\), called Datalog\(^+\).

Let us denote with \(L\) a language that consists of all the possible Datalog clauses built from a finite number of predicates. We distinguish two subsets of \(L\), namely \(L_o\) and \(L_h\), which are the language of observations and the language of hypotheses, respectively. Shortly, the model inference problem can be stated as follows. Given two sets of ground facts \(E^+\) (positive examples) and \(E\) (negative examples), expressed in the language of observations \(L_o\) and a background knowledge \(B\) in the language \(L\) (in our setting, \(B\) is bound to be a set of ground atoms), the model inference problem consists in finding a logic program \(P\) (theory) such that \(P \cup B \vdash E^+\) (completeness) and \(P \cup B \not\vdash E\) (consistency).

In the literature of machine learning, it is possible to find many examples of algorithms that solve model inference problems. They can be roughly subdivided into batch and incremental, according to the fact that training examples \(- E^+ \cup E -\) are completely available at learning time or not, respectively. For instance, MIS is an incremental algorithm based on the Popperian methodology of conjectures and refutations [26]. The importance of this algorithm is not limited to the area of machine learning, but extends to algorithmic debugging of logic programs.

MIS is able to infer a logic theory from a sequence of examples by modifying incrementally a conjecture (a logic program) whenever a contradiction occurs between the conjectured theory and the given examples, that is to say, whenever the current theory is either not complete (too weak) or not consistent (too strong). Part of MIS is the