Extending incomplete information sources

Carlo Meghini and Yannis Tzitzikas
Consiglio Nazionale delle Ricerche
Istituto della Scienza e delle Tecnologie della Informazione
Pisa, Italy

Nicolas Spyratos
Université Paris-Sud
Laboratoire de Recherche en Informatique
91405 Orsay Cedex, France

Abstract

The extraction of information from a source containing term-classified objects is plagued with uncertainty, due, among other things, to the possible incompleteness of the source index. To overcome this incompleteness, the study proposes to extend the index of the source, in a way that is as reasonable as possible with respect to the original classification of objects. By equating reasonableness with logical implication, the sought expansion turns out to be an explanation of the index, captured by abduction. We study the general problem of query evaluation on the extended information source, providing a polynomial time algorithm which tackles the general case, in which no hypothesis is made on the structure of the taxonomy. We then specialize the algorithm for two well-know structures: DAGs and trees, showing that each specialization results in a more efficient query evaluation.

1 Introduction

Taxonomies is probably the oldest and most widely used conceptual modeling tool. Nevertheless, it is a powerful tool still used is in Web directories (e.g. in Google and Yahoo!), Content Management (hierarchical structures are used frequently to classify documents), Web Services (services are typically classified in a hierarchical form), Marketplaces (goods are classified in hierarchical catalogs), Personal File Systems, Personal Bookmarks for the Web, Libraries (e.g. Thesauri [10]) and in very large collections of objects (e.g. see [19]). Although more sophisticated conceptual models (including concepts, attributes, relations and axioms) have emerged and are recently employed even for meta-tagging in the Web [12, 24], almost all of them have a backbone consisting of a subsumption hierarchy, i.e. a taxonomy. Furthermore, the design of taxonomies can be done more systematically if done following a faceted approach (e.g. see [18, 17]) and thanks to techniques that have emerged recently [20], taxonomies of compound terms can be also defined in a flexible and systematic manner. Moreover, the advantages of the taxonomy-based conceptual modeling approach for building large scale P2P systems that support semantic-based retrieval services have been analyzed and reported in [23, 22].

The extraction of information from an information source (hereafter, IS) containing term-classified objects is plagued with uncertainty. From the one hand, the indexing of objects, that is the assignment of a set of terms to each object, presents many difficulties, whether it is performed manually by some expert or automatically by a computer programme. In the former case, subjectivity may

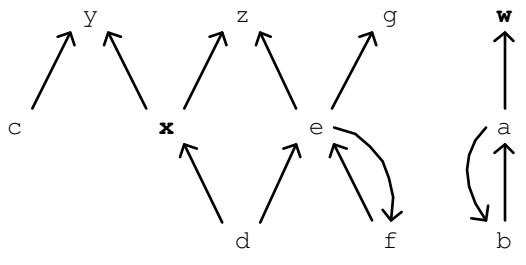
play a negative role (e.g. see [25]); in the latter case, automatic classification methods may at best produce approximations. On the other hand, the query formulation process, being linguistic in nature, would require perfect attuning of the system and the user language, an assumption that simply does not hold in open settings such as the Web.

A collection of textual documents accessed by users via natural language queries is clearly a kind of IS, where documents play the role of objects and words play the role of terms. In this context, the above mentioned uncertainty is typically dealt with in a quantitative way, i.e. by means of numerical methods: in a document index, each term is assigned a *weight,* expressing the extent to which the document is deemed to be about the term. The same treatment is applied to each user query, producing an index of the query which is a formal representation of the user information need of the same kind as that of each document. Document and query term indexes are then matched against each other in order to estimate the relevance of the document to a query (e.g. see [1]).

In the present study, we take a different approach, and deal with uncertainty in a *qualitative* way. We view an IS as an agent, operating according to an open world philosophy. The agent knows some facts, but it does not interpret these facts as the only ones that hold; the agent is somewhat aware that there could be other facts, compatible with the known ones, that might hold as well, although they are not captured for lack of knowledge. These facts are, indeed, *possibilities.* One way of defining precisely in logical terms the notion of possibility, is to equate it with the notion of *explanation.* That is, the set of terms associated to an object is viewed as a *manifestation* of a phenomenon, the indexing process, for which we wish to find an explanation, justifying why the index itself has come to be the way it is. In logic, the reasoning required to infer explanations from given theory and observations, is known as *abduction.* We will therefore resort to abduction in order to define precisely the possibilities that we want our system to be able to handle. In particular, we will define an operation that extends an IS by adding to it a set (term, object) pairs capturing the sought possibilities, and then study the property of this operation from a mathematical point of view. The introduced operation can be used also for ordering query answers using a *possibility*-based measure of relevance.

One could say that the IS's we assume resemble those Information Retrieval Systems (IRSs) which employ the boolean retrieval model (see [1] for a review) and exploit lexical ontologies or word-based thesauri [10] (like WordNet [3] and Roget's thesaurus) for query expansion, i.e. for expanding queries with synonyms, hyponyms and related terms in order to improve recall (e.g. see [16], [13], [14], [9]). However note that the IRS techniques are applicable only if the objects of the domain have a textual content (this is not a prerequisite of our approach). Another remarkable difference with our sources is that the taxonomies employed by IRSs usually do not accept the semantic interpretation that we describe in this paper. Lexical ontologies like WordNet [3] are structured using lexical relations (synonymy, hyponymy, antonymy) which are not semantic relations. For instance, according to Wordnet, *window* is subsumed by *opening* and by *panel.* However, every *window* is not a *panel* and an *opening,* thus extensional subsumption does not hold here (for more about this problem see [8]).

The paper is structured as follows. Next Section introduces the basic notion of the study, that of an information source, as well as the mechanism of query-based information extraction. Section 3 contains the main technical developments: it presents the extension of an information source by means of an abduction-based operator, as well as sound and complete algorithms to achieve this extension. The complexity of query evaluation on extended information sources is also derived. Section 4 studies the process of iterating the extension operator from a mathematical point of view, showing convergence to a fixed point. In Section 5, two important types of information sources are studied, namely those whose taxonomy is shaped as a DAG (named hierarchical) and



$$
\begin{aligned}
T &= \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}, \mathtt{e}, \mathtt{f}, \mathtt{g}, \mathtt{w}, \mathtt{x}, \mathtt{y}, \mathtt{z}\} \\
K &= \{\mathtt{c} \to \mathtt{x},\ \mathtt{x} \to \mathtt{y},\ \mathtt{x} \to \mathtt{z},\ \mathtt{d} \to \mathtt{x},\ \mathtt{d} \to \mathtt{e}, \\
&\quad\ \mathtt{e} \to \mathtt{z},\ \mathtt{e} \to \mathtt{g},\ \mathtt{e} \to \mathtt{f},\ \mathtt{f} \to \mathtt{e},\ \mathtt{a} \to \mathtt{w}, \\
&\quad\ \mathtt{a} \to \mathtt{b},\ \mathtt{b} \to \mathtt{a}\} \\
Obj &= \{1\} \\
I &= \{(\mathtt{x}, 1),\ (\mathtt{w}, 1)\}
\end{aligned}
$$

Figure 1: An information source

as a tree, respectively. It is shown that query evaluation on an extended information sources is more tractable for these two classes of information sources. Finally, Sections 6 and 7 elaborate on further applications of the framework previously developed. Specifically, Section 6 shows how to use abduction to rank the results of a query, while Section 7 discusses several applications of the extensions operator.
A preliminary, short version of this work has been reported in [15].

2 Information Sources

An information source consists of two basic elements: a taxonomy and a structure on it.

Definition 1: An *information source* (IS) S is a pair $S = (O, U)$ where

- O, the *taxonomy,* is a pair $O = (T, K)$ where T is a finite set of symbols, called the *terms* of the taxonomy, and K is a finite set of conditionals on T, *i.e.* formulas of the form $p \to q$ where p and q are different terms of the taxonomy;
- U is a *structure on* O, that is a pair $U = (Obj, I)$ where Obj is a countable set of objects, called the *domain* of the structure, and I is a finite relation from T to Obj, that is $I \subseteq T \times Obj$.

K is called the *knowledge base* of the taxonomy, while I is called the *interpretation* of the structure. □

As customary, we will treat the relation I as a function from terms to sets of objects and, where t is a term in T, sometimes write $I(t)$ to denote the set $I(t) = \{o \in Obj \mid (t, o) \in I\}$. We will call $I(t)$ the *extension* of term t. Dually, given an object $o \in Obj$, the *index of* o *in* S, $ind_S(o)$, is given by the set of terms which have o in their extension:

$$ind_S(o) = \{t \in T \mid (t, o) \in I\}.$$

Finally, the *context of* o *in* S, $C_S(o)$, is defined as: $C_S(o) = ind_S(o) \cup K$. For any object o, $C_S(o)$ consists of terms and simple conditionals that collectively form all the knowledge about o that S has.

Example 1: Throughout the paper, we will use as an example the IS $S = ((T, K), (Obj, I))$ given in the righthand side of Figure 1. The lefthand side of the Figure graphically illustrates the taxonomy of S and its interpretation, by showing in boldface the terms in the index of the only object, *i.e.*, $ind_S(1) = \{\mathtt{x}, \mathtt{w}\}$. □

In this study, we focus on ISs which satisfy an intuitive minimality criterion. In order to introduce this criterion and keep the paper self-contained, a few basic notions from propositional logic [6]

are now recalled. Given a set of propositional variables P, a *truth assignment for* P is a function mapping P to the true and false truth values, respectively denoted by **T** and **F**. A truth assignment V *satisfies* a sentence σ of the propositional calculus (PC), $V \models \sigma$, if σ is true in V, according to the classical truth valuation rules of PC. A set of sentences Σ *logically implies* the sentence α, $\Sigma \models \alpha$, iff every truth assignment which satisfies every sentence in Σ also satisfies α.
The *instance set* of an object o in an IS S, denoted as $N_S(o)$, is the set of terms that are logically implied by the context of o in S

$$N_S(o) = \{t \in T \mid C_S(o) \models t\}.$$

For each term t in $N_S(o)$, we will say that o is an *instance* of t. Clearly, $ind_S(o) \subseteq N_S(o)$, therefore o is an instance of each term in $ind_S(o)$.

Definition 2: The index of an object o in an IS S, $ind_S(o)$, is *non-redundant* iff

$$A \subset ind_S(o) \text{ implies } \{v \in T \mid A \cup K \models v\} \subset N_S(o).$$

An IS is *non-redundant* if all its indices are non-redundant. □

In practice, the index of an object is non-redundant if no term in it can be removed without loss of information. It can be easily verified that the IS introduced in the previous example is non-redundant. From now on, we will consider "IS" as a synonym of "non-redundant IS".

2.1 Querying Information Sources

We next introduce the query language for extracting information from an IS in the traditional question-answering way.

Definition 3: Given a taxonomy $O = (T, K)$, the *query language for* O, $\mathcal{L}_O$, is defined by the following grammar, where t is a term in T:

$$q ::= t \mid q \wedge q' \mid q \vee q' \mid \neg q \mid (q)$$

Any expression in $\mathcal{L}_O$ is termed a *query.* □

The answer to queries is defined in terms of satisfaction in the model of an object context, the *truth model,* realizing a closed-world reading of an IS.

Definition 4: Given an IS $S = (O, U)$, for every object $o \in Obj$, the *truth model of* o *in* S, $V_{o,S}$, is the truth assignment for T defined as follows, for each term $t \in T$:

$$V_{o,S}(t) = \begin{cases} \mathbf{T} & \text{if } C_S(o) \models t \\ \mathbf{F} & \text{otherwise} \end{cases}$$

Given a query φ in $\mathcal{L}_O$, the *answer of* φ *in* S is the set of objects whose truth model satisfies the query:

$$ans(\varphi, S) = \{o \in Obj \mid V_{o,S} \models \varphi\}.$$

□

In the Boolean model of information retrieval, a document is returned in response to a query if the index of the document satisfies the query. Thus, the above definition extends Boolean retrieval by considering also the knowledge base in the retrieval process.

t	a	b	c	d	e	f	g	x	y	z	w
$V_{1,S}(t)$	**F**	**F**	**F**	**F**	**F**	**F**	**F**	**T**	**T**	**T**	**T**

Table I: The truth model of object 1 in the information source S

Example 2: Table I shows the truth model of object 1 in the IS introduced in Example 1. The answer to the query z in the same IS, $ans(\mathtt{z}, S)$, consists of object 1, since $V_{1,S}(\mathtt{z}) = \mathbf{T}$. Instead, $ans(\mathtt{c}, S) = ans(\mathtt{a}, S) = \emptyset$. □

Following Definition 4, query evaluation requires the computation of the truth model of each object o, which in turn requires deciding whether each query term is logically implied by the object context $C_S(o)$. Computing propositional logical implication is in general a difficult task. However, the specific form of the propositional theories considered in this study, makes this computation much simpler, as the remainder of this Section shows. In order to devise an efficient query evaluation procedure, we will resort to graph theoretic concepts.
The *term graph* of a taxonomy O is the directed graph $G_O = (T, E)$, such that $(t, t') \in E$ iff $t \rightarrow t'$ is in K. Figure 1 shows indeed the term graph of the example IS. For simplicity, we will use "term" also to refer to a vertex of the term graph. The *tail of a term* t in G_O, *tail(t),* is the set of terms that can be reached from t by walking the graph edges backward, that is:

$$tail(t) = \{u \in T \mid \text{ there exists a path from } u \text{ to } t \text{ in } G_O\}$$

Proposition 1: For all ISs S and queries $\varphi \in \mathcal{L}_O$, $ans(\varphi, S) = \alpha_S(\varphi)$, where α_S is the *solver* of the IS S, defined as follows:

$$\begin{aligned} \alpha_S(t) &= \bigcup \{I(u) \mid u \in tail(t)\} \\ \alpha_S(q \wedge q') &= \alpha_S(q) \cap \alpha_S(q') \\ \alpha_S(q \vee q') &= \alpha_S(q) \cup \alpha_S(q') \\ \alpha_S(\neg q) &= Obj \setminus \alpha_S(q) \end{aligned}$$

Proof: By structural induction on the query φ. But we first need to establish the following:

Lemma 1: Given an IS S, a set of terms $A \subseteq T$ and a term $t \in T$, $A \cup K \models t$ iff there is a path in G_O from a letter in A to t.
Proof: ($\rightarrow$) By induction on the length l of the path. For $l = 0$, the path is (t, t) which means that $t \in A$, so $A \models t$ and by the monotonicity of $\models$, $A \cup K \models t$. Suppose the Proposition holds for $l = k > 0$. Let $\langle u_1, \ldots, u_k, t\rangle$ be a path in G_O with $u_1 \in A$, and M be a model of $A \cup K$. By definition of G_O, $u_k \rightarrow t$ is in K. Since M is a model of $A \cup K$ and by the induction hypothesis, $M(u_k) = \mathbf{T}$, it follows that $M(t) = \mathbf{T}$. Hence M is a model of t, so $A \cup K \models t$.
($\leftarrow$) Suppose $A \cup K \models t$ yet G_O contains no path from a letter in A to t. Let M be the interpretation defined as follows:

$$M(v) = \begin{cases} \mathbf{T} & \text{if there is a path from a letter in } A \text{ to } v \text{ in } G_O \\ \mathbf{F} & \text{otherwise} \end{cases}$$

By definition, any letter in A is $\mathbf{T}$ in M, so M is a model of A. Let $p \rightarrow q$ be a conditional in K. If $M(p)$ is $\mathbf{F}$, then the conditional is true in M. If, on the other hand, $M(p) = \mathbf{T}$, then there is a path from a letter u in A to p; since G_O contains the edge (p, q), the path from u to p can be extended to q, and by definition $M(q) = \mathbf{T}$. Also in this case

M satisfies the conditional. So M also satisfies every conditional in K, hence M is a model of $A \cup K$. Since $A \cup K \models t$, M is also a model of t, that is $M(t) = \mathbf{T}$. However, by construction, $M(t) = \mathbf{F}$, a contradiction. □

Case $\varphi = t$. By definition, $ans(t, S) = \{o \in Obj \mid V_{o,S} \models t\}$. $V_{o,S} \models t$ iff $C_S(o) \models t$ iff $ind_S(o) \cup K \models t$ iff (by the previous Lemma) there is a path in G_O from a letter $u \in ind_S(o)$ to t, which is the same to say that $o \in I(u)$ for some $u \in tail(t)$. So, $ans(t, S) = \{o \in Obj \mid \exists u \in tail(t)$ such that $o \in I(u)\}$, therefore

$$ans(o, S) = \cup\{I(u) \mid u \in tail(t)\}.$$

Case $\varphi = q \wedge q'$. $ans(q \wedge q', S) = \{o \in Obj \mid V_{o,S} \models q \wedge q'\} = \{o \in Obj \mid V_{o,S} \models q$ and $V_{o,S} \models q'\}$ $= \{o \in Obj \mid V_{o,S} \models q\} \cap \{o \in Obj \mid V_{o,S} \models q'\} = ans(q, S) \cap ans(q', S)$.
The other cases are analogous. □

Example 3: In the IS previously introduced, the term `z` can be reached in the term graph by each of the following terms: `z`, `x`, `d`, `e`, `f`. Hence, $tail(\mathtt{z}) = \{\mathtt{z}, \mathtt{x}, \mathtt{d}, \mathtt{e}, \mathtt{f}\}$. According to the last Proposition, then: $ans(\mathtt{z}, S) = \alpha_S(\mathtt{z}) = I(\mathtt{z}) \cup I(\mathtt{x}) \cup I(\mathtt{d}) \cup I(\mathtt{e}) \cup I(\mathtt{f}) = \{1\}$. □

Proposition 2: $\alpha_S(t)$ can be computed in $O(|T| \cdot |Obj| \cdot log\,|Obj|)$ time.
Proof: From the last Proposition, computing $\alpha_S(t)$ requires the following steps: (a) to derive *tail(t)* by searching the term graph in order to identify every vertex that is backward reachable from t; (b) to access the extension of each term in *tail(t)*; and (c) to compute the union of the involved extensions. The time complexity of step (a) is $|T|^2$, corresponding to the case in which every term is backward reachable from t in the term graph. We assume that step (b) can be performed in constant time, which is negligible with respect to the other values at stake. Let us now consider step (c). By adopting a merge-sort strategy, the union between two extensions can be performed in $n\ log\, n$ time in the size of the input. Since in the worst case the union of $|T|$ extensions must be computed, and each extension is the whole set Obj, we have a $O(|T| \cdot |Obj| \cdot log\,|Obj|)$ time complexity for step (c). Overall, the upper bound for evaluating single-term queries is therefore $O(|T|^2 + |T| \cdot |Obj| \cdot log\,|Obj|)$. Since the size of the domain is expected to be significantly larger than the size of the terminology, the sought upper bound for singles-term queries evaluation is $O(|T| \cdot |Obj| \cdot log\,|Obj|)$. □

3 Extended Information Sources

Let us suppose that a user has issued a query against an IS and is not satisfied with the answer, as the answer does not contain objects that are relevant to the user information need. Further, let us assume that the user is not willing to replace the current query with another one, for instance because of lack of knowledge on the available language or taxonomy. In this type of situation, database systems offer practically no support, as they are based on the assumption that users can always articulate their information need in the form of a query. In an information retrieval setting a user in the above described situation could use relevance feedback to pinpoint interesting (relevant) or uninteresting objects among those returned by the system, and ask the system to re-evaluate the query taking into account this information; but what if all the displayed objects are not relevant? Assuming that the IS indeed contains relevant objects, the user's disappointment could be due to a linguistic mismatch between the user and the indexer. That is, the latter attributes to the IS terms a different meaning, or a different usage, than the former, and, as a result, object classification is inconsistent with the user expectation. For instance, in the IS introduced in Example 1, what the user would call an `x`, the system calls a `z`. Maybe the concept of an `x` is vague, and in presence of

a borderline case the user would take a liberal approach, and classify it as an `x`, while the system could show a more conservative attitude and classify it as `z`.
On the other hand, even if the user and the indexer were attuned to the same language, a situation as the one described above could occur due to a mistake of the indexer, who failed to recognize an object as, *e.g.* an `x`. Or, `x` and `z` have been created to refine a previously existing term t, and in distributing t's extension to `x` and `z` something went wrong. Given the widespread usage of automatic indexers based on statistical techniques, these cases would not seem very unlikely to occur. Or, there could be operational misbehavior: for instance, the term `x` could have been added to the system in the recent past, and it is not yet fully exploited in the index of the IS.
In all these, and probably other, cases, the index of the IS suffers from *incompleteness:* it contains correct information, but at least in some cases not *all* the correct information. In other words, there are other facts, compatible with the known ones, which hold as well, although they are not captured for *lack of knowledge.*
To overcome this lack of knowledge, the idea is to relax the index, by expanding it, in a way that is as *reasonable* as possible with respect to the original classification of objects. But what could be a reasonable expansion? By reasonable expansion we mean a *logically grounded* expansion, that is an expansion that *logically implies* the index as it has been created in the first place. Then, the expansion we are talking about is in fact a logical *explanation* of the index. The most general form of explanation in logic is *abduction,* seen as the generation of causes to explain the observed effects of known laws. In our case, the known laws are the sentences of the IS knowledge base, the observed effects are contents of the index, while the cause is the sought index expansion. We will therefore resort to abduction in order to define precisely the expansion that would address the incompleteness of the index.

3.1 Propositional abduction problems

The model of abduction that we adopt is the one presented in [5], now briefly recalled for self-containedness. Let $\mathcal{L}_V$ be the language of propositional logic over a finite alphabet V of propositional variables. A *propositional abduction problem* is a tuple $\mathcal{A} = \langle V, H, M, Th \rangle$, where $H \subseteq V$ is the set of hypotheses, $M \subseteq V$ is the manifestation, and $Th \subseteq \mathcal{L}_V$ is a consistent theory. $S \subseteq H$ is a solution (or explanation) for $\mathcal{A}$ iff $Th \cup S$ is consistent and $Th \cup S \models M$. $Sol(\mathcal{A})$ denotes the set of the solutions to $\mathcal{A}$.
In the context of an IS S, we will consider each object separately. Thus,

- the terms in T play both the role of the propositional variables V and of the hypotheses H, as there is no reason to exclude *apriori* any term from an explanation;
- the knowledge base K plays the role of the theory Th;
- the role of manifestation is played by the index of the object.

Consequently, we have the following:

Definition 5: Given an IS $S = (O, U)$ and object $o \in Obj$, the *propositional abduction problem for o in S*, $\mathcal{A}_S(o)$, is the propositional abduction problem $\mathcal{A}_S(o) = \langle T, T, ind_S(o), K \rangle$. □

The solutions to $\mathcal{A}_S(o)$ are given by:

$$Sol(\mathcal{A}_S(o)) = \{A \subseteq T \mid K \cup A \models ind_S(o)\}$$

where the consistency requirement on $K \cup A$ has been omitted since for no knowledge base K and set of terms A, $K \cup A$ can be inconsistent. Usually, certain explanations are preferable to others, a

fact that is formalized in [5] by defining a preference relation $\preceq$ over $Sol(\mathcal{A})$. Letting $a \prec b$ stand for $a \preceq b$ and $b \not\preceq a$, the set of preferred solutions is given by:

$$Sol_{\preceq}(\mathcal{A}) = \{S \in Sol(\mathcal{A}) \mid \nexists S' \in Sol(\mathcal{A}) : S' \prec S\}.$$

Also in the present context a preference relation is desirable, satisfying criteria that reflect the goals of our framework. Here are these criteria, in order of decreasing importance:

1. explanations including only terms in the manifestation are less preferable, as they do not provide any additional information;

2. explanations altering the behavior of the IS to a minimal extent are preferable; this requirement acts in the opposite direction of the previous one, by making preferable solutions that, if incorporated in the IS, minimize the differences in behavior between the so extended IS and the original one;

3. between two explanations that alter the behavior of the IS equally, the simpler one is to be preferred. As explanations are sets, it is natural to equate simplicity with smallness in size.

All the above criteria can be expressed in terms of the effects produced by the extension of an IS. In order to indicate these effects, we will use the term "perturbation", defined next.

Definition 6: Given an IS $S = (O, U)$, an object $o \in Obj$ and a set of terms $A \subseteq T$, the *perturbation of A on S with respect to o*, $pert_o(A)$ is given by:

$$pert_o(A) = \{t \in T \mid (C_S(o) \cup A) \models t \text{ and } C_S(o) \not\models t\}$$

that is the set of additional terms in the instance set of o once the index of o is extended with the terms in A. □

We can now define the preference relation over solutions of the above stated abduction problem.

Definition 7: Given an IS $S = (O, U)$, an object $o \in Obj$ and two solutions A and A' to the problem $\mathcal{A}_S(o)$, $A \preceq A'$ if either of the following holds:

1. $pert_o(A') = \emptyset$
2. $0 < |pert_o(A)| < |pert_o(A')|$
3. $0 < |pert_o(A)| = |pert_o(A')|$ and $A \subseteq A'$. □

The strict correspondence between the clauses in the last Definition and the criteria previously set for the preference relation should be evident. Solutions having an empty perturbation are obviously subsets of the instance set of the object, therefore the first condition of the last Definition captures the first of the three criteria. The second condition establishes preference for solutions that minimize the number of terms that change their truth value from **F** to **T** in the truth model of the object, and thus alter the behavior of the IS *with respect to query answering* to a minimal extent. Between two solutions producing the same alteration, the third condition makes preferable the smaller in size, and so simplicity, criterion number three, is implemented.
We now introduce the notion of *extension* of an IS. The idea is that an extended IS (EIS for short) includes, for each object, the terms of the original index plus those added through the abduction process illustrated above. However, in so doing, non-redundancy may be compromised, since no constraint is posed on the solutions of the abduction problem in order to avoid it. There can be two sources of redundancy when extending a non-redundant index with the solutions to an abduction problem:

1. As it will be shown later (Proposition 0), a solution to an abduction problem may contain taxonomical cycles, and including a whole cycle in the index of an object clearly violates non-redundancy (all terms in the cycle but one can be removed without losing information).

2. A term in a solution may be a direct descendant of a term in the index. The coexistence of these two terms in the new index violates redundancy, thus the latter can be added only if the former is removed.

In order to cope with the former problem, we introduce the operator ρ, which takes as input a set of terms and replaces each cycle occurring in it by any term in the cycle. In order to cope with the latter problem, we introduce a special union operator $\sqcup$ which takes as input two interpretations and adds each pair (t, o) of the second interpretation to the first interpretation after removing any pair (u, o) in the first interpretation such that $t \rightarrow u \in K$. Formally,

$$I_1 \sqcup I_2 = I_2 \cup \{(t, o) \in I_1 \mid \text{for no pair } (v, o) \in I_2,\ v \rightarrow t \in K\}.$$

Definition 8: Given an IS $S = (O, U)$ and an object $o \in Obj$, the *abduced index* of o, $abind_S(o)$, is given by:

$$abind_S(o) = \bigcup Sol_{\preceq}(\mathcal{A}_S(o)) \setminus ind_S(o).$$

The *abduced interpretation of* S, I^+, is given by

$$I^+ = I \sqcup \{\langle t, o\rangle \mid o \in Obj \text{ and } t \in \rho(abind_S(o))\}.$$

Finally, the *extended* IS, S^e, is given by $S^e = (O, U^e)$ where $U^e = (Obj, I^+)$. □

3.2 Querying extended information sources

A key role in solving propositional abduction problems is played by single-letter solutions (SLSs, for short) introduced next.

Definition 9: Given an IS S, an object o and a term $t \in T \setminus N_S(o)$, the *single-letter solution of* t is the set $\mu(t)$ given by:

$$\mu(t) = \{t\} \cup (ind_S(o) \setminus \sigma(t))$$

where $\sigma(t) = \{u \in T \mid \text{there is a path in } G_O \text{ from } t \text{ to } u\}$. □

The next Lemma states some useful properties of SLSs.

Lemma 2: For any IS S, object $o \in Obj$ and term $t \in T$:

1. $\mu(t)$ is a solution to $\mathcal{A}_S(o)$

2. $pert_o(\mu(t)) = \sigma(t) \setminus N_S(o)$

3. if $t \notin N_S(o)$, $\mu(t)$ has the smallest perturbation among the solutions to $\mathcal{A}_S(o)$ including t

4. for any letter $v \in T \setminus N_S(o)$, $\{v\} \cup K \models t$ implies $pert_o(\mu(t)) \subseteq pert_o(\mu(v))$.

Proof: (1) By definition of $\models$, $(ind_S(o) \setminus \sigma(t)) \models (ind_S(o) \setminus \sigma(t))$. By Lemma 1 and the definition of $\sigma(t)$, $\{t\} \cup K \models \sigma(t)$. By combining the last two relationships, we have:

$$(ind_S(o) \setminus \sigma(t)) \cup \{t\} \cup K \models (ind_S(o) \setminus \sigma(t)) \cup \sigma(t) \supseteq ind_S(o).$$

t	$\sigma(t)$	$\mu(t)$	$pert_o(\mu(t))$
a	{w, a, b}	{a, x}	{a, b}
b	{w, a, b}	{b, x}	{a, b}
c	{c, y}	{c, x, w}	{c}
d	{d, x, y, z, e, f, g}	{d, w}	{d, e, f, g}
e	{e, f, z, g}	{e, x, w}	{e, f, g}
f	{f, e, z, g}	{f, x, w}	{e, f, g}
g	{g}	{g, x, w}	{g}

Table II: The single-letter solutions of $\mathcal{A}_S(1)$ and their perturbations

Hence $\mu(t) \cup K \models ind_S(o)$, and therefore $\mu(t) \in Sol(\mathcal{A}_S(o))$.
(2) We have:

$$\begin{aligned} pert_o(\mu(t)) &= \{u \in T \mid (C_S(o) \cup \mu(t)) \models u \text{ and } C_S(o) \not\models u\} \\ &= \{u \in T \mid (ind_S(o) \cup K \cup \{t\} \cup (ind_S(o) \setminus \sigma(t)) \models u\} \setminus N_S(o) \\ &= \{u \in T \mid (ind_S(o) \cup K \cup \{t\}) \models u\} \setminus N_S(o) \\ &= \{u \in T \mid (K \cup \{t\}) \models u\} \setminus N_S(o) \\ &= \sigma(t) \setminus N_S(o) \end{aligned}$$

(3) Let $A \in Sol(\mathcal{A}_S(o))$, $A \neq \mu(t)$, such that $t \in A$. It follows that $\{t\} \subseteq A$. We must prove that $pert_o(\mu(t)) \subseteq pert_o(A)$. It can be verified that, for any set of terms $X \subseteq T$, $pert_o(X) = pert_o(X \setminus N_S(o))$, and that $pert_o$ is monotonic in $T \setminus N_S(o)$, that is $A, A' \subseteq T \setminus N_S(o)$ and $A \subseteq A'$ imply $pert_o(A) \subseteq pert_o(A')$. Putting all together,

$$pert_o(\mu(t)) = pert_o(\mu(t) \setminus N_S(o)) = pert_o(\{t\}) \subseteq pert_o(A)$$

(4) There can be two cases: (1) $t \in N_S(o)$. Then, $pert_o(\mu(t)) = \emptyset \subseteq pert_o(\mu(v))$. (2) $t \notin N_S(o)$. Then, by the hypothesis, $t \in \sigma(v)$, therefore $\sigma(t) \subseteq \sigma(v)$. It follows $\sigma(t) \setminus N_S(o) \subseteq \sigma(v) \setminus N_S(o)$, so by the monotonicity of $pert_o$ in $T \setminus N_S(o)$, $pert_o(\mu(t)) \subseteq pert_o(\mu(v))$. □

Example 4: Let us consider again the IS S introduced in Example 1, and the problem $\mathcal{A}_S(1)$. The manifestation is given by $ind_S(1) = \{\mathtt{w}, \mathtt{x}\}$ while $N_S(1) = \{\mathtt{x}, \mathtt{y}, \mathtt{z}, \mathtt{w}\}$. Table II gives, for each term in $T \setminus N_S(1)$, the σ value, the single-letter solution and its perturbation. □

We can now state the main result on query answering in EIS.

Proposition 3: For all ISs S and terms $t \in T$,

$$ans(t, S^e) = \alpha_{S^e}(t) = \alpha_S(t) \cup \{o \in Obj \mid t \in abind_S(o)\}.$$

Proof: By definition, $ans(t, S^e) = \{o \in Obj \mid V_{o,S^e} \models t\}$. $V_{o,S^e} \models t$ iff $C_{S^e}(o) \models t$ iff $ind_{S^e}(o) \cup K \models t$ iff $\{t \in T \mid (t, o) \in I^+\} \cup K \models t$ iff $\{t \in T \mid (t, o) \in I \text{ or } t \in abind_S(o)\} \cup K \models t$ iff $ind_S(o) \cup abind_S(o) \cup K \models t$. Hence, $ans(t, S^e) = \{o \in Obj \mid ind_S(o) \cup K \models t\} \cup \{o \in Obj \mid abind_S(o) \cup K \models t\} = \alpha_S(t) \cup \{o \in Obj \mid abind_S(o) \cup K \models t\}$. We now prove that

$$\alpha_S(t) \cup \{o \in Obj \mid abind_S(o) \cup K \models t\} = \alpha_S(t) \cup \{o \in Obj \mid t \in abind_S(o)\}.$$

We first show that the lefthand side is contained in the righthand side. This is obvious if $o \in \alpha_S(t)$. Let o be such that $abind_S(o) \cup K \models t$. By Lemma 1, for some term $v \in T$, $v \in abind_S(o)$ and there exists a path in G_O from v to t. If $v = t$, then $t \in abind_S(o)$ and the Lemma holds. Now let us

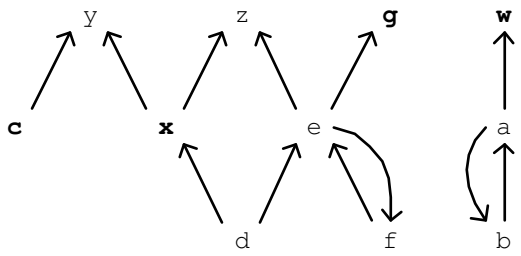


Figure 2: An extended information source

suppose $v \neq t$. There can be two cases: (1) $v \in N_S(o)$. By definition and Lemma 1, there exists a path in G_O from a letter $w \in ind_S(o)$ to v. But then there is a path in G_O from w to t, and by the same Lemma, $ind_S(o) \cup K \models t$. This implies $o \in \alpha_S(t)$. (2) $v \notin N_S(o)$. If $t \in N_S(o)$, $o \in \alpha_S(t)$, so let us assume $t \notin N_S(o)$. Since $v \in abind_S(o)$, by Lemma 2.(3) $\mu(v) \in Sol_{\preceq}(\mathcal{A}_S(o))$, hence $|pert_o(\mu(v))| \leq |pert_o(\mu(t))|$. Since $\{v\} \cup K \models t$, by Lemma 2.(4) $|pert_o(\mu(v))| \geq |pert_o(\mu(t))|$. It follows that $|pert_o(\mu(v))| = |pert_o(\mu(t))|$ and therefore $\mu(t) \in Sol_{\preceq}(\mathcal{A}_S(o))$ too. By definition of $abind_S(o)$ and since $t \notin N_S(o)$, $t \in abind_S(o)$.
We now show that the righthand side is contained in the lefthand side. Again, this is obvious if $o \in \alpha_S(t)$. Let o be such that $t \in abind_S(o)$. Then, $abind_S(o) \models t$. By the monotonicity of $\models$, $abind_S(o) \cup K \models t$. □

The last Proposition indicates that, in order to answer queries against an EIS, one needs to compute the set

$$\beta(t) = \{o \in Obj \mid t \in abind_S(o)\}.$$

The properties of SLSs stated in Lemma 2 allows us to derive $abind_S(o)$ for any propositional abduction problem.

Proposition 4: Given an IS S and an object $o \in Obj$, let d_o be the least positive perturbation of the solutions to $\mathcal{A}_S(o)$, that is: $d_o = min\{|pert_o(A)| \mid A \in Sol(\mathcal{A}_S(o)) \text{ and } pert_o(A) > 0\}$. Then

$$abind_s(o) = \{t \in T \setminus N_S(o) \mid |\sigma(t) \setminus N_S(o)| = d_o\}$$

Proof: ($\rightarrow$) We first show that the lefthand side is contained in the righthand side. By Definition of $abind_S(o)$, $t \in abind_S(o)$ implies $t \notin N_S(o)$. Moreover, it implies that there exists a solution A in $Sol(\mathcal{A}_S(o))$ containing t, such that $|pert_o(A)| = d_o$. By Lemma 2.(3), $A = \mu(t)$. By Lemma 2.(2), $pert_o(\mu(t)) = \sigma(t) \setminus N_S(o)$, hence $|\sigma(t) \setminus N_S(o)| = d_o$.
($\leftarrow$)By Lemma 2.(1), $\mu(t) \in Sol(\mathcal{A}_S(o))$. By the hypothesis $|\sigma(t) \setminus N_S(o)| = d_o$ and Lemma 2.(2), it follows that $pert_o(\mu(t)) = d_o$ and therefore $\mu(t) \in Sol_{\preceq}(\mathcal{A}_S(o))$. By the hypothesis $t \notin N_S(o)$ and the definition of $abind_S(o)$, $t \in abind_S(o)$. □

Example 5: Let us consider again the problem $\mathcal{A}_S(1)$ of the last Example. From Table II and the last Proposition, it follows that $abind_S(1) = \{\mathtt{c}, \mathtt{g}\}$ and that $I^+ = I \cup \{(\mathtt{c}, 1), (\mathtt{g}, 1)\}$. The resulting EIS is shown in Figure 2, following the same conventions used in Figure 1. □

Proposition 4 suggests the following algorithm for computing $\beta(t)$: for each object $o \in Obj$, select the letters u in $T \setminus N_S(o)$ that minimize the size of $\sigma(u) \setminus N_S(o)$. If t is amongst these letters, $o \in \beta(t)$, otherwise $o \notin \beta(t)$. Procedure $\beta 1$, shown in Figure 3, implements this algorithm. $\beta 1$ uses *beta*, N_S, d, $ABIND$ and π to compute, respectively, $\beta(t)$, $N_S(o)$, the minimum size of $\pi(u)$, $abind_S(o)$ and $\pi(u)$. *beta* is initialized to the empty set, then the main loop is entered. In it,

```
procedure β1 (S : IS ; t : term ; beta : set of objects)
 1.  begin
 2.  beta ← ∅
 3.  for each object o ∈ Obj in S do
 4.     begin
 5.     N_S ← succ(ind_S(o))
 6.     d ← |T| + 1
 7.     ABIND ← ∅
 8.     for each term u ∈ T \ N_S do
 9.        begin
10.        π ← succ({u}) \ N_S
11.        if |π| < d then
12.           begin
13.           d ← |π|
14.           ABIND ← π
15.           end
16.        else if |π| = d then ABIND ← ABIND ∪ π
17.        end
18.     if  t ∈ ABIND then beta ← beta ∪ {o}
19.     end
20. end
```

Figure 3: The $\beta 1$ procedure

for each object $o \in Obj$, $\beta 1$ first computes $N_S(o)$ by the *succ* function. This function visits the successors in G_O of each element in the input set, and is not reported as it implements an algorithm in elementary graph theory which can be found on any textbook. Then, d is initialized to $|T| + 1$, which is greater than any value d may assume, and so is equivalent to infinite, while $ABIND$ is initialized to the empty set. For each letter $u \in T \setminus N_S(o)$, $\beta 1$ computes in π the set $\sigma(u) \setminus N_S(o)$ and collects in $ABIND$ the letters which minimize the size of π. Finally, only if t is amongst these letters, $\beta 1$ adds o to *beta*. Soundness and completeness of $\beta 1$ directly follow from Proposition 4, and allows us to obtain an upper bound on the complexity of computing $\beta(t)$.

Proposition 5: $\alpha_{S^e}(t)$ can be computed in $O(|T| \cdot |Obj|^2 \cdot log\,|Obj|)$ time.
Proof: $\beta 1$ loops on Obj, and, for each object o, it first loops on $T \setminus N_S(o)$ (lines 8-17) and then updates the variable *beta* (line 18). Let us consider the loop on $T \setminus N_S(o)$. In the worst case, $N_S(o)$ is empty, so that this loop is executed $|T|$ times. At each iteration, $\beta 1$ invokes the *succ* function, which visits the term graph in order to find the terms reachable from the current term. The worst case is when the term graph is fully connected, so that, at each iteration, the number of edges to be visited equals $|T|^2$, and the variable π is assigned the whole terminology. In this case, d is always equal to $|T|$ and the test at line 16 is always passed, except at the first iteration, when the test at line 11 is passed. The time complexity of each iteration of the loop, different from the first, is then given by $|T|^2 + |Obj|\,log\,|Obj|$, and that of the entire loop is $O(|T|^3 + |T| \cdot |Obj|\,log\,|Obj|)$. The test on line 18 is always passed in the worst case, and every object o is inserted into *beta*, which grows by one element at each iteration, until it equals Obj. By keeping β ordered, this insertion has time complexity $log\,k$, where k is the current size of *beta*. It can be proved that the dominant term in

$$\sum_{j=1}^{|Obj|} log\,j$$

is $|Obj|\ log\,|Obj|$, thus the overall complexity of $\beta 1$ is $O(|Obj| \cdot |T|^3 + |T| \cdot |Obj|^2 \cdot log\,|Obj|)$. Since, as already remarked, the size of the terminology is expected to be significantly smaller than that

of the domain, the upper bound on $\beta(t)$ is $O(|T| \cdot |Obj|^2 \cdot log\,|Obj|)$. By combining the fact that $\alpha_S(t) \cup \beta(t)$ and Proposition 2 , it follows that $O(|T| \cdot |Obj|^2 \cdot log\,|Obj|)$ is also an upper bound on $\alpha_{S^e}(t)$. □

The last results establishes that query evaluation on EISs worsens complexity by a factor equals to the size of the domain, and is consistent with the complexity results derived from propositional abduction problems [5]. The remaining of this Section is devoted to devise a method for computing $\beta(t)$ that is in practice more efficient than the rather naive one sketched in the proof of the last Proposition. The framework developed to this end will be useful in establishing complexity upper bounds for special kinds of ISs (Section 5).

3.3 Minimal terminal sets

We first observe that we do not need to compute $abind_S(o)$, *i.e.* solve an abduction problem, for each object $o \in Obj$. Objects having the same index give rise to the same propositional abduction problem and therefore have the same abduced index. Then, we only need to solve an abduction problem for each different index. Let $\approx$ be a binary relation on objects, defined as follows:

$$o \approx o' \text{ if and only if } ind_S(o) = ind_S(o').$$

$\approx$ is an equivalence relation. Let $[o]$ be the equivalence class of object o. We call each $[o]$ a *class*. The above consideration is captured in the following Proposition:

Proposition 6: For all ISs S and terms $t \in T$, $\beta(t) = \cup\{\ [o] \mid t \in abind_S(o)\}$.
Proof: For each object $o' \in [o]$, $o' \in \beta(t)$ iff $t \in abind_S(o')$. But $ind_S(o) = ind_S(o')$ implies $abind_S(o) = abind_S(o')$. Hence $t \in abind_S(o')$ iff $t \in abind_S(o)$. □

In the worst case, the number of classes equals the number of objects, but in the average case, the former is expected to be much smaller than the latter, so considering classes should result in a sensible optimization.
Another major source of inefficiency of algorithm $\beta 1$ is that it loops on the set $T \setminus N_S(o)$. This set can be very large, and computing the perturbation of each single-letter solution for each term in it, may be very time consuming. Fortunately, as it will be shown soon, $\beta(t)$ can be computed by exploring only a part of the set $T \setminus N_S(o)$, by visiting (a subgraph of) the term graph in the appropriate way.

Definition 10: Given a graph $G = (V, E)$, a *terminal set in* G is either a singleton $\{v\}$, where v is a vertex with no outgoing edges, or is a set of vertices $\{v_1, v_2, \ldots, v_k\}$, $k \geq 2$, such that $E(v_i) = \{v_{i+_k 1}\}$ for all $1 \leq i \leq k$, where $+_k$ is sum modulo k. A terminal set A in G is *minimal* if no other terminal set A' in G exists such that $|A'| < |A|$. Any element of a minimal terminal set is called a *minterm*. □

We can now state the basic result on which the method for answering queries on EIS relies. Given a graph $G = (V, E)$ and a set $V' \subseteq V$, the subgraph of G induced by V' is the graph $G' = (V', E')$, where $E' = \{(u, v) \in E \mid u, v \in V'\}$.

Proposition 7: Given an IS S, an object o and a term $t \in T$, $t \in abind_S(o)$ iff t is a minterm in the subgraph G^o_O of the term graph G_O induced by $T \setminus N_S(o)$.
Proof: ($\rightarrow$) $t \in abind_S(o)$ implies that $\mu(t)$ has least positive perturbation. Suppose t is not a minterm in G^o_O. Then there exists a term $v \in T \setminus N_S(o)$ such that $(t, u) \in E$ and $(u, t) \notin E$. But then $pert_o(\mu(v)) < pert_o(\mu(t))$, a contradiction.
($\leftarrow$) Let t be a minterm. There are two cases: (1) $\{t\}$ is a minimal terminal set. In this case,

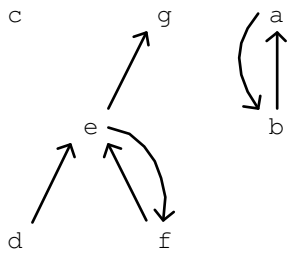


Figure 4: The graph G^1_O

$pert_o(\mu(t)) = 1$ and there cannot be a smaller positive perturbation in $Sol(\mathcal{A}_S(o))$; so $\mu(t) \in Sol_{\preceq}(\mathcal{A}_S(o))$ and $t \in abind_S(o)$. (2) $A = \{v_1, v_2, \ldots, v_k\}$ is a minimal terminal set and $t = v_j$ for some $1 \leq j \leq k$. It follows that $p(S, o\mu(v_1)) = \ldots = p(S, o, \nu(v_k)) = k$. By the minimality of A no smaller terminal set exists. Then, for all $1 \leq i \leq k$, $\mu(v_i) \in Sol_{\preceq}(\mathcal{A}_S(o))$, hence $v_i \in abind_S(o)$, and in particular $t \in abind_S(o)$. □

Example 6: Figure 4 shows G^1_O, that is the subgraph of the term graph in Figure 1 induced by $N_S(1)$. G^1_O has 3 terminal sets: $\{\mathtt{g}\}$, $\{\mathtt{c}\}$ and $\{\mathtt{a}, \mathtt{b}\}$, the first two of which are minimal. Therefore $abind_S(1) = \{\mathtt{c}, \mathtt{g}\}$. □

The last Proposition reduces the computation of $\beta(t)$ to that of the minterms in the graph G^o_O. In general, G^o_O will have two kinds of terminal sets:

- the terminal sets in G_O not involving terms in $N_S(o)$; we call these *genuine* terminal sets, and
- the terminal sets created by the pruning of G_O giving rise to G^o_O; we call these *induced* terminal sets.

We assume that the terminal sets of the term graph are computed at system start-up, then stored, and revised every time the term graph is modified. Since they play a crucial role in answering queries on EIS, we think this is a safe assumption, from a database point of view. Then, genuine terminal sets are found by identifying the stored terminal sets which are free from elements in $N_S(o)$. Induced terminal sets can be found by exploring the borders of $N_S(o)$ in G_O as done by procedure $\beta 2$, presented in Figure 5.
$\beta 2$ takes as input an IS S and a term t, and computes, in *beta,* $\beta(t)$. It operates under the following assumptions, for a given IS S,

1. the classes of S are known;
2. the components $G_1, \ldots, G_{n_S}$ of the term graph G_O are known;
3. G_O is represented through adjacency sets, *i.e.* for each term $t \in T$, $Adj[t]$ is a set containing all terms u such that there is an edge from t to u, *i.e.* $(t, u) \in E$;
4. inverse adjacency sets are also available, *i.e.* for each term $t \in T$, $Adj^{-1}(t)$ gives the nodes $u \in T$ such that $(u, t) \in E$.

$\beta 2$ uses the same variables as $\beta 1$. It loops on the classes of the IS S, and for each such class $[o]$, it computes in N_S the set $N_S(o)$ via the *succ* function. Then, a loop is entered on each component G_i of the term graph G_O; within this loop, genuine terminal sets are first identified, by checking

```
procedure β2 (S : IS ; t : term ; beta : set of objects)
 1.  begin
 2.  beta ← ∅
 3.  for each class [o] in S do begin
 4.      N_S ← succ(ind_S(o))
 5.      d ← |T| + 1
 6.      ABIND ← ∅
 7.      for each component G_i = (T_i, E_i) of G_O do begin
 8.          for each terminal set TS of G_i do
 9.              if TS ∩ N_S = ∅ then
10.                  if |TS| < d then begin
11.                      d ← |TS|
12.                      ABIND ← TS
13.                      end
14.                  else if |TS| = d then ABIND ← ABIND ∪ TS
15.          B ← ∅
16.          for each term t ∈ N_S do
17.              for each term u ∈ Adj^{-1}[t] \ N_S do
18.                  if u ∉ B then begin
19.                      B ← B ∪ {u}
20.                      if Adj[u] ⊆ N_S then
21.                          if d > 1 then begin
22.                              d ← 1
23.                              ABIND ← {u}
24.                              end
25.                          else ABIND ← ABIND ∪ {u}
26.                      else explore(⟨u⟩)
27.                      end
28.          end
29.      if  t ∈ ABIND then beta ← beta ∪ [o]
30.      end
31.  end

procedure explore (path : sequence of terms)
 1.  begin
 2.  t ← last(path)
 3.  A ← Adj[t] \ N_S
 4.  u ← get(A)
 5.  if |A| = 1 then
 6.      if u ∈ path then
 7.          if |path| < d then
 8.              begin
 9.              d ← |path|
10.              ABIND ← path
11.              end
12.          else if |path| = d then ABIND ← ABIND ∪ path
13.      else if u ∉ B and Adj[u] ⊈ N_S then
14.          begin
15.          B ← B ∪ {u}
16.          explore(append(path, u))
17.          end
18.  end
```

Figure 5: The $\beta 2$ and *explore* procedures

each stored terminal set of the component for intersection with N_S (line 9). If the terminal set TS does not contain any term in N_S, then it is a potential minimal terminal set, thus, its cardinality is compared to the current value of d. If the former is smaller than the latter, then TS is smaller than any terminal set found so far, therefore $ABIND$ is set to it, and d is set to its cardinality (lines 11 and 12). If the cardinality of TS is the same as the current value of d, then TS is another minimal solution according to the current d, and as such it is added to $ABIND$ (line 14). If neither is the case, TS is just ignored as not minimal. $\beta 2$ then proceeds to compute induced terminal sets. To this end, it needs to visit the term graph. The set B is used to remember the terms already visited; initially, B is empty (line 15). A loop is then entered, in which, for each term t in N_S, each predecessor u not in N_S is considered (line 17). If u is not yet visited (line 18), it is inserted in B and considered in the present iteration. There can be two cases:

1. All direct successors of u, $Adj[u]$, are in N_S (line 20). In this case, $\{u\}$ is a terminal set in G^o_O, which implies that the least positive perturbation d_o is 1. Hence, if the current value of d is greater than 1, all solutions found so far are not minimal: d is set to 1 and $ABIND$ is initialized with $\{u\}$ (lines 22 and 23). If, on the other hand, the current value of d is 1, then u is simply added to $ABIND$ (line 25).

2. There is some direct successors of u not in N_S. In this case, u may still be a minterm, in case it belongs to a terminal set with more than one element. In order to check whether this is the case, the procedure *explore* is invoked, with the sequence consisting of just u as input (line 26).

When all the minterms of the current class have been identified, $\beta 2$ checks whether t is amongst these. If yes (line 29), then $[o]$ is added to *beta*.
The procedure *explore*, also presented in Figure 5, takes as input a sequence *path* of terms outside N_S that is a path in G^o_O; each element of this path, except the last one, has been found to have the next element in *path* as its only successor outside N_S, as required by Definition 10. The last element in *path* is known to have at least one successor outside N_S. *path* is thus a potential terminal set, and an effective one just in case *path* will at some point become a cycle, consisting of elements having exactly one successor outside N_S. The procedure *explore* explores the graph G_O in order to make sure it detects such a terminal set, if one exists. To this end, it focuses on the last element of *path,* t, which is also the first one, upon the first invocation. The variable A is assigned the successors of t outside N_S (line 3), and u is anyone of these (line 4). If t has more than one successor outside N_S, *i.e.* $|A| > 1$, it violates the condition of Definition 10requiring that each element in a terminal set have exactly one successor, so *explore* is exited with no further action. If, on the other hand, $|A| = 1$, then there can be two cases:

1. u is an element in *path.* If yes, then a cycle has been detected, and an induced terminal set. Consequently, if the size of the cycle is smaller than the current d (line 7), all current solutions are not minimal, so both d and $ABIND$ are re-initialized, respectively with the size of *path* and with its elements (lines 9 and 10). If, instead, d equals the size of *path*, then *path* is just added to $ABIND$ (line 11).

2. u is not already in *path.* In this case, u may have been already visited, *i.e.* $u \in B$, which means that *path* cannot be a cycle, otherwise all its element would be in B. Or, being newly discovered, u may have no successor at all, which means that it is a genuine terminal set, or may have all successors in N_S, which means that it is an induced terminal set. If it is a genuine terminal set, then it has already been identified in the first stage of $\beta 2$ and need not be considered; if it is an induced terminal set, then, being not already visited, will be

identified as soon as one of its successors in N_S is considered. Therefore, in all these cases the current *path* is no solution, and no action needs be done. The only interesting case is if u is not visited and has some successor outside N_S. In this case, *explore* adds u to the visited terms and recursively invokes itself on the current path extended with u (line 16).

Soundness and completeness of the $\beta 2$ procedure directly follow form the above considerations and from Proposition 7.

4 Iterative Extension of Information Sources

The operator $\cdot^+$ extends the index of an IS based on the notion of explanation, logically captured by abduction. Once this mechanism is in place, it is natural to envisage a more thorough exploitation of it, for instance taking the form of an iterative application of the extension operator $\cdot\ +$. As an alternative, one may think of a more substantial extension, based on a less strict preferential criterion amongst solutions to the involved abduction problem; but for the present context we will consider only the former kind of exploitation.
Intuitively, we would expect that $\cdot^+$ be a function which, applied to an interpretation, produces an equal or a larger interpretation, the former case corresponding to the situation in which the knowledge base of the IS does not enable to find any explanations for each object index. Technically, this amounts to say that $\cdot^+$ is a monotonic function, which is in fact the case. Then, by iterating the $\cdot^+$ operator, we expect to move from an interpretation to a larger one, until an interpretation is reached which cannot be extended any more. Also this turns out to be true, and in order to show it, we will model the domain of the $\cdot^+$ operator as a complete partial order, and use the notion of fixed point in order to capture interpretations that are no longer extensible.

Proposition 8: Given an IS S, let the *domain* of S be the set $\mathcal{D}$ given by $\mathcal{D} = \{I \cup A \mid A \in \mathcal{P}(T \times Obj)\}$. Then, $\cdot^+$ is a continuous function on the complete partial order $(\mathcal{D}, \subseteq)$.
Proof: $(\mathcal{D}, \subseteq)$ is obviously a partial order; moreover, it has I as least element, while the least upper bound of every chain K in $\mathcal{D}$ is $\cup K$. This proves that $(\mathcal{D}, \subseteq)$ is a complete partial order. We next show that $\cdot^+$ is monotonic, that is, that given any two IS S_1 and S_2 with the same taxonomy (T, K) and interpretations $I_1, I_2 \in \mathcal{D}$, respectively, $I_1 \subseteq I_2$ implies $I_1^+ \subseteq I_2^+$. Let $(t, o) \in I_1^+$. We must prove that $(t, o) \in I_2^+$. There are two cases. (1) $(t, o) \in I_1$. By the hypothesis, $(t, o) \in I_2$ and by definition of I_2^+, $(t, o) \in I_2^+$. (2) $(t, o) \notin I_1$. If $(t, o) \in I_2$ then again by definition of I_2^+, $(t, o) \in I_2^+$. So, let us assume $(t, o) \notin I_2$. Then, $t \in abind_{S_1}(o)$ and therefore $\mu(t) \cup K \models ind_{S_1}(o)$. By hypothesis, $ind_{S_1}(o) \subseteq ind_{S_2}(o)$, hence $\mu(t) \cup K \models ind_{S_2}(o)$, and therefore $\mu(t)$ is a solution of the abduction problem $\mathcal{A}_{S_2}(o)$. Now, from $ind_{S_1}(o) \subseteq ind_{S_2}(o)$, it follows that $N_{S_1}(o) \subseteq N_{S_2}(o)$, and therefore the perturbation of any single-letter solution $\mu(a)$ in S_1, $\sigma(a) \setminus N_{S_1}(o)$, is going to be larger than that in S_2, given by $\sigma(a) \setminus N_{S_1}(o)$. This implies that $\mu(t)$ has minimal perturbation in S_2 too. So, $t \in abind_{S_2}(o)$. If t has been selected by the function ρ to be in I_1^+, it will also be selected by the same function to be in I_2^+, hence $(t, o) \in I_2^+$ and the monotonicity of $\cdot^+$ is proved. Its continuity follows from monotonicity the fact that in the considered complete partial order all chains are finite, hence the class of monotonic functions coincides with the class of continuous functions (see, *e.g.* [7]). □

As a corollary of the previous Proposition and of the Knaster-Tarski fixed point theorem, we have that the function $\cdot^+$ has a least fixed point that is the least upper bound of the chain $\{I, I^+, (I^+)^+, \ldots\}$. As already remarked, all chains are finite, so the fixed point will be reached after a finite number of iterations. According to Proposition 4, for each object o, the terms added at each iteration are amongst those in $T \setminus N_S(o)$. It follows that the fixed point is reached when, for

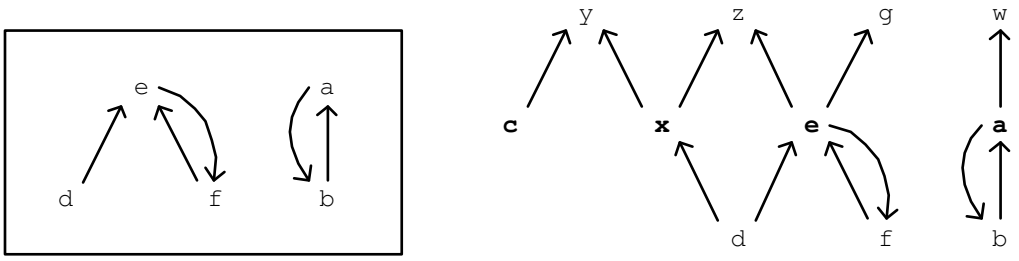


Figure 6: Extending the information source P

each object o, $N_S(o) = T$. This guarantees that the extension process always covers the entire terminology, if pushed to its extreme consequences. Moreover, the same Proposition allows to derive an upper bound on the number of iterations required to reach the fixed point. In fact, assuming that at each iteration $abind_S(o)$ consists only of one term, in at most

$$max\{\ |T \setminus N_S(o)|\ |\ o \in Obj\}$$

iterations the fixed point is reached.

Example 7: Let us consider the IS $S^e = P$ shown in Figure 2 and the problem $\mathcal{A}_P(1)$. The new graph G^1_O is shown, framed, in the lefthand side of Figure 6. This graph has two terminal sets $\{\mathtt{a}, \mathtt{b}\}$ and $\{\mathtt{e}, \mathtt{f}\}$, both minimal. It follows that $abind_P(1) = \{\mathtt{a}, \mathtt{b}, \mathtt{e}, \mathtt{f}\}$. The function ρ, applied to this set, excludes one element of each cycle, so to obtain, for instance, $\rho(abind_P(1)) = \{\mathtt{a}, \mathtt{e}\}$. The abduced interpretation of P, following Definition 8, is then given by:

$$\{(\mathtt{c}, 1), (\mathtt{x}, 1), (\mathtt{g}, 1), (\mathtt{w}, 1)\} \sqcup \{(\mathtt{a}, 1), (\mathtt{e}, 1)\}$$

The pair $(\mathtt{a}, 1)$ "rules out" the pair $(\mathtt{w}, 1)$, while $(\mathtt{e}, 1)$ rules out $(\mathtt{g}, 1)$. The abduced index is therefore:

$$\{(\mathtt{c}, 1), (\mathtt{x}, 1), (\mathtt{e}, 1), (\mathtt{a}, 1)\}.$$

Let Q be the IS so obtained. If we further solve the problem $\mathcal{A}_Q(1)$, the new graph G^1_O consists just of the vertex $\mathtt{d}$, and the only minimal terminal set is $\{\mathtt{d}\}$, which is also $abind_Q(1)$. The abduced index of Q is given by:

$$\{(\mathtt{c}, 1), (\mathtt{x}, 1), (\mathtt{e}, 1), (\mathtt{a}, 1)\} \sqcup \{(\mathtt{d}, 1)\} = \{(\mathtt{c}, 1), (\mathtt{d}, 1), (\mathtt{a}, 1)\}.$$

Q^e is shown in Figure 7. At this point, $N_{Q^e}(1) = T$, so the interpretation of Q^e is a fixed point and no further extension is possible. □

5 Special Information Sources

We conclude this study by applying the ideas developed so far to two special classes of ISs, corresponding to two special structures of the taxonomy. The first class consists of the ISs whose term graphs are directed acyclic graphs (DAGs) with a maximal element $\top$. We call these ISs "hierarchical". Indeed, hierarchical taxonomies are common in object-oriented models, where subsumption is a partial ordering relation amongst classes and maximal elements are introduced in order to tie classes up, thus making each class reachable from the top. It will be shown that the evaluation of queries on extended hierarchical ISs is a much simpler problem than the general one, studied in the

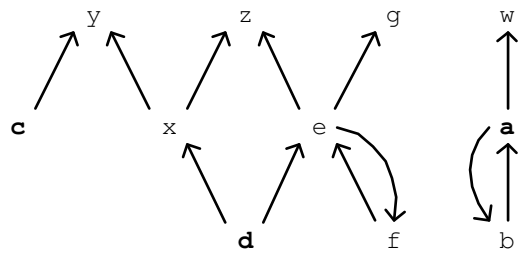


Figure 7: Extending the information source Q

previous part of the paper. The second class includes a special case of the first, that is term graphs that are rooted trees. This kind of taxonomies are common in catalogs, directories and information retrieval systems on the Web, and, as expected, they make query evaluation on extended ISs even simpler.
The following Proposition addresses query evaluation on extended hierarchical ISs.

Proposition 9: Let a *hierarchical* information source (HIS) be an IS whose term graph is a DAG with a greatest element $\top$. Then, for all HISs S and terms $t \in T$, $t \neq \top$,

$$ans(t, S^e) = \cap \{\alpha_S(u) \mid t \rightarrow u \in K\}.$$

Proof: From Proposition 7, a term $t \in abind_S(o)$ iff t is a minterm in the subgraph G^o_O. Since the term graph G^o_O of a HIS is a DAG, G^o_O is a DAG too, hence its terminal sets, according to Definition 10, are those singletons $\{u\}$ such that u is a term with no outgoing edges. Each such terminal set is obviously minimal, being a singleton. Thus, an object $o \in \beta(t)$ iff t has no outgoing edges in G^o_O. There may be two cases. (1) $N_S(o) = \emptyset$. In this case $G_O = G^o_O$, and $\top$ is the only minterm. Thus $t = \top$ and the Proposition holds vacuously. (2) $N_S(o) \neq \emptyset$. In this case, since $\top$ is the greatest element of the taxonomy, $\top \in N_S(o)$, so $t \neq \top$. Then, t is a minterm iff all immediate successors of t in G_O are in $N_S(o)$, but t is not. In this way, when $N_S(o)$ is "erased" from G_O to produce G^o_O, t is left with no outgoing edges. It follows that:

$$\begin{aligned}
\beta(t) &= \{o \in Obj \mid t \notin N_S(o), \text{ and } t \rightarrow u \in K \text{ implies } u \in N_S(o)\} \\
&= \{o \in Obj \mid C_S(o) \not\models t, \text{ and } t \rightarrow u \in K \text{ implies } C_S(o) \models u\} \\
&= ans(\bigwedge\{u \mid t \rightarrow u \in K\} \wedge \neg t, S) \\
&= \cap \{\alpha_S(u) \mid t \rightarrow u \in K\} \setminus \alpha_S(t)
\end{aligned}$$

From Proposition 3, $ans(t, S^e) = \alpha_S(t) \cup \beta(t)$, therefore $\alpha_S(t) \cup (\cap \{\alpha_S(u) \mid t \rightarrow u \in K\} \setminus \alpha_S(t))$ and the Proposition follows. □

The last Proposition establishes that an object o is in the result of query t against the EIS S^e just in case it is an instance of all the immediate generalizations of t in S. Indeed, if o were an instance of t, it would, as a consequence, be an instance of all t's generalizations, thus the explanation of the current index offered by the system is the most reasonable one can conceive. As a result, the behavior of the query mechanism turns out to be compliant with intuition. Notice that asking the query $\top$ on the extended IS, a case not dealt with by the last Proposition, does not make much sense, since already $ans(\top, S) = Obj$.

Example 8: Let us consider the HIS S shown in Figure 8, where, as usual, boldface is used to indicate the index of the only object, 1, of the IS. The problem $\mathcal{A}_S(1)$ has two minimal solutions,

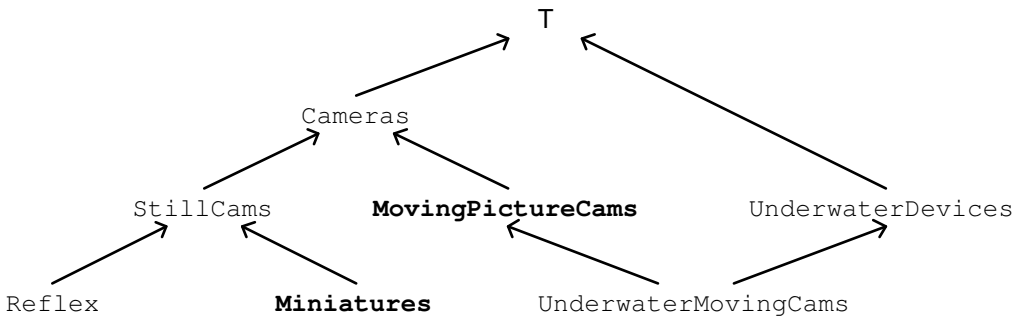


Figure 8: A hierarchical IS.

given by $\mu(\texttt{Reflex})$ and $\mu(\texttt{MovingPictureCams})$. Indeed, `Reflex` and `MovingPictureCams` are the minterms of the graph G^1_O, which includes only the term `UnderwaterMovingCams`, beside these. Thus, object 1 should be returned in response to the query `Reflex` on S^e, and in fact:

$$\begin{aligned} ans(\texttt{Reflex}, S^e) &= \cap \{\alpha_S(u) \mid \texttt{Reflex} \rightarrow u \in K\} = \alpha_S(\texttt{StillCams}) \\ &= I(\texttt{StillCams}) \cup I(\texttt{Miniatures}) \cup I(\texttt{Reflex}) = \{1\} \end{aligned}$$

Letting P stand for S^e and considering $\mathcal{A}_P(1)$, the only minterm is now `UnderwaterMovingCams`), thus 1 should be included in the answer to the query `UnderwaterMovingCams` on P^e. Indeed:

$$\begin{aligned} ans(\texttt{UnderwaterMovingCams}, P^e) &= \cap \{\alpha_P(u) \mid \texttt{UnderwaterMovingCams} \rightarrow u \in K\} \\ &= \alpha_P(\texttt{MovingPictureCams}) \cap \alpha_P(\texttt{UnderwaterDevices} = \{1\} \end{aligned}$$

since 1 is an instance of `MovingPictureCams` in S and has become an instance of `UnderwaterDevices` in P. □

From a complexity point of view, the Proposition permits to compute an upper bound on the evaluation of queries on an extended HIS.

Proposition 10: $\alpha_{S^e}(t)$, where S is a HIS, can be computed in $O(|T|^2 \cdot |Obj| \cdot log\,|Obj|)$ time.
Proof: The query evaluation procedure derived from the last Proposition consists of 3 steps: (a) to compute each immediate successors u of t in the term graph; (b) to evaluate the query u on the HIS; (c) to take the intersection of the results. In order to derive the worst case, let x be the number of terms that are immediate successors of t, and y the terms that are reachable backward from each immediate successor of t. In the worst case, $y = |T| - x$. Using the result established in Proposition 2 for the complexity of step (b), the cost function has a maximum for $x = \frac{|T|-1}{2}$. Using this value, step (a) requires $\frac{|T|-1}{2}$ time; step (b) $O(|T|^2 \cdot |Obj| \cdot log\,|Obj|)$; step (c) $O(|T| \cdot |Obj| \cdot log\,|Obj|)$. The cost of step (b) is clearly the largest of the three, so the Proposition follows. □

Table III summarizes the complexity results obtained for query evaluation on the classes of IS examined in this study. The last result indicates that the evaluation of queries on extended HISs is worse than that on ISs by a factor proportional to the size of the terminology. This is a significant improvement over the general case, where the factor is proportional to the size of the domain (Proposition 5). This difference reflects the fact that in the general case, $\alpha_{S^e}(t)$ must be computed in an object-based (or class-based) fashion, *i.e.* by considering one object (class) at a time, while the evaluation of $\alpha_{S^e}(t)$ on a HIS proceeds in a term-based fashion, *i.e.* by considering the terms that are immediate successors of t in the term graph. This also simplifies the implementation, as it avoids to compute and keep track of classes.

IS type	Complexity
Simple	$O(\|T\| \cdot \|Obj\| \cdot log\,\|Obj\|)$
Extended	$O(\|T\| \cdot \|Obj\|^2 \cdot log\,\|Obj\|)$
Extended Hierarchical	$O(\|T\|^2 \cdot \|Obj\| \cdot log\,\|Obj\|)$
Extended Tree	$O(\|T\| \cdot \|Obj\| \cdot log\,\|Obj\|)$

Table III: Summary of complexity results for query evaluation

We now proceed to address the second special class of IS.

Proposition 11: Let a *tree* information source (TIS) be an IS whose term graph is a rooted tree with root $\top$. Then, for all TISs S and terms $t \in T$, $t \neq \top$,

$$ans(t, S^e) = \{\alpha_S(u) \mid t \rightarrow u \in K\}.$$

Proof: A TIS is a HIS in which every term different from $\top$ has exactly one immediate successor in the term graph. □

The complexity of query evaluation on extended TISs is clearly the same as that on ISs.

6 Ranked answers

At the beginning of Section 3 a number of scenarios have been introduced in order to motivate the following abduction-based IS extension. In this Section and in the next, we further elaborate on the possible usage of the IS extension mechanism.

The abduction framework described can be also exploited for obtaining ordered answers. In order to illustrate how, let us use a superscript to indicate the iteration at which an IS is generated, that is, $S = S^0$, $S^e = S^1$, $(S^e)^e = S^2$ and so on. Moreover, let N be the iteration at which the fixed point is reached, *i.e.* $S^{N-1} \subset S^N = S^{N+1} = S^{N+2} = \ldots$. The set of objects that the user will get in response to a query φ on the extensions of the IS S, is given by:

$$answer_S(\varphi) = \bigcup_{i=0}^{N} \alpha_{S^i}(\varphi)$$

We can give all of these objects to the user as a response to the query φ on S, ordered according to the iteration at which each object would start appearing in the answer. In particular, we can define the *rank* of an object $o \in answer_S(\varphi)$, denoted by $rank_S(o, \varphi)$, as follows:

$$rank_S(o, \varphi) = min\{\ k \mid o \in \alpha_{S^k}(\varphi)\}$$

The answer that will be returned by φ on S, the *ranked answer*, is an ordering of sets, i.e. the ordering:

$$rans(\varphi, S) = \langle \{o \mid rank_S(o, \varphi) = 1\}, \{o \mid rank_S(o, \varphi) = 2\}, ..., \{o \mid rank_S(o, \varphi) = N\} \rangle$$

For example, consider the hierarchical IS presented in Figure 9, where the extension of each term is shown on the right of the term (*i.e.*, $I(\texttt{MovingPictureCams}) = \{2, 3\}$). Let suppose that $\varphi = \texttt{UMC}$. In this case we have:

$$\begin{aligned} \alpha_{S^0}(\texttt{UMC}) &= \emptyset \\ \alpha_{S^1}(\texttt{UMC}) &= \{3\} \\ \alpha_{S^2}(\texttt{UMC}) &= \{1, 2, 3\} \end{aligned}$$

so the ranked answer to UMC is $rans(\texttt{UMC}, S) = \langle \{3\}, \{1, 2\} \rangle$.

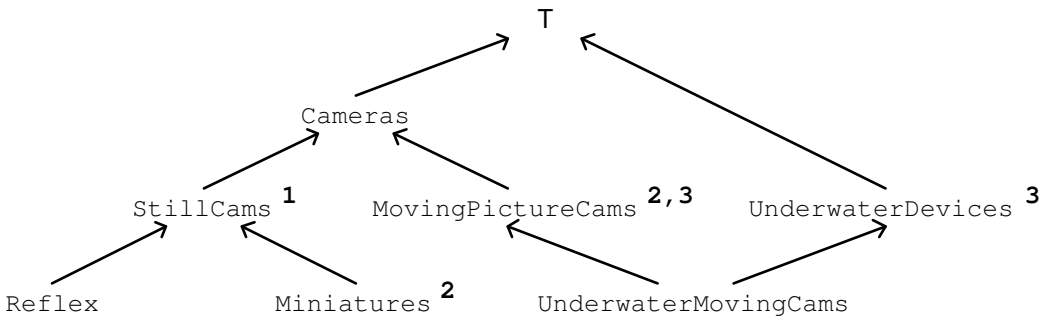


Figure 9: A hierarchical IS.

7 Applications

The proposed approach can be applied on collections of manually indexed objects (e.g. Web Catalogs), as well as on automatically indexed collections of texts (whose index is derived using automatic classification methods). The only prerequisite is that the employed taxonomy should admit of a semantic interpretation such as the one given in this paper. Specifically the proposed operation can be applied to taxonomies whose relationships can be seen as conditionals. Taxonomies that fall to this category include:

- Taxonomies which conceptualize the domain as a denumerable set of objects, and consist of terms that denote sets of objects and relationships that denote extensional subsumption[1]. The subject hierarchies of Web catalogs fit in this case.
- Taxonomies which consist of terms that denote geographical areas structured by a spatial inclusion relation (e.g. `Crete` $\rightarrow$ `Greece`).
- Taxonomies (for products, living species, fields of knowledge) in which the terms correspond to categories structured by a subclass relation (e.g. `Canaries` $\rightarrow$ `Birds`). For example many existing thesauri fit in this case.

It can also be applied on secondary (i.e. mediator) taxonomy-based sources [21] for alleviating not only the uncertainty that originates from primary sources, but also the uncertainty and inexactness of the mediator mappings.
Also notice that by restricting the set of solutions A in Definition 5 we can limit the scope of the extension operation to the parts of the index that we want to relax. This restriction can be applied on taxonomies designed according to a faceted approach [18, 17] in order to relax the index with respect to the desired facet(s).
Another interesting remark is that the proposed operation can be applied not only to manually constructed taxonomies but also to taxonomies derived automatically on the basis of an inference service. For instance, it can be applied on sources indexed using taxonomies of *compound terms* which are defined algebraically [20]. Furthermore it can be applied on concept lattices formed using Description Logics (DL) [4]. Specifically, we can view a DL knowledge base $\Sigma = (TBox, ABox)$ as a source $S_\Sigma = (O, U)$ where the taxonomy $O = (T, K)$ and the structure $U = (Obj, I)$ are as follows:

- T consists of all concepts that appear in $TBox$ and $ABox$;

[1] In contrast to the intensional meaning of terms (i.e. see [11], [2]) and to intensional subsumption.

– K contains a conditional $C \rightarrow D$ $(C, D \in T)$ iff $\Sigma \models C \sqsubseteq D$, *i.e.* C subsumes D in the knowledge base Σ;

– Obj is the set of individuals that appear in the concept assertions of $ABox$; and

– I is defined by the concept assertions of the $ABox$, namely $(C, o) \in I$ iff $C(o) \in Abox$.

For example consider a query Q posed to the knowledge base Σ. Let $S_{\Sigma \cup Q}$ be the source obtained by adding the query Q to the source S_Σ as defined earlier, i.e. by adding Q to T and placing Q in its correct place (with respect to $\sqsubseteq$) in K. The extension operator $\cdot^+$ can then be applied to the source $S_{\Sigma \cup q}$.

8 Conclusions

Indexing accuracy and consistency are difficult to maintain. It has been known for a long time that there are large variations between the index terms assigned by different indexers, or by the same indexer at different times [25]. Imposing a standard indexing language (like the terminology of a taxonomy) tends to improve consistency, but it does not improve accuracy. The same holds for the automatic classification methods which may at best produce approximations. To alleviate this problem we have proposed a mechanism which allows liberating the index of a source in a gradual manner. This mechanism is governed by the notion of explanation, logically captured by abduction.

The proposed method can be implemented as an answer enlargement[2] process where the user is not required to give additional input, but from expressing his/her desire for more objects. The introduced framework can be also applied for ranking the objects of an answer according to an explanation-based measure of relevance. Recall that in statistics-based Information Retrieval the notion of relevance is founded only experimentally; what relevance is, in other words, is defined by the user from time to time and from experiment to experiment, and is then heavily dependent on judgments where highly subjective and hardly reproducible factors are brought to bear. Our approach can be used for ordering the objects of the answer according to a more well-founded notion of relevance.

References

[1] R. Baeza-Yates and B. Ribeiro-Neto. *"Modern Information Retrieval"*. ACM Press, Addison-Wesley, 1999.

[2] Mario Bunge. *Scientific Research I. The Search for Systems*, 3(1), 1967.

[3] Princeton University Cognitive Science Laboratory. "WordNet: A Lexical Database for the English Language". (http://www.cogsci.princeton.edu/ wn).

[4] F.M. Donini, M. Lenzerini, D. Nardi, and A. Schaerf. Reasoning in description logics. In G. Brewka, editor, *Principles of Knowledge Representation*, Studies in Logic, Language and Information, pages 193–238. CSLI Publications, 1996.

[5] T. Eiter and G. Gottlob. The complexity of logic-based abduction. *Journal of the ACM*, 42(1):3–42, January 1995.

[2] If the query contains negation then the answer can be reduced.

[6] H.B. Enderton. *A mathematical introduction to logic*. Academic Press, N. Y., 1972.

[7] P.A. Fejer and D.A. Simovici. *Mathematical Foundations of Computer Science. Volume 1: Sets, Relations, and Induction*. Springer-Verlag, 1991.

[8] Nicola Guarino. "Some Ontological Principles for Designing Upper Level Lexical Resources". In *Proceedings of first int. Conf. on Language Resources and Evaluation*, Granada, Spain, May 1998.

[9] Nicola Guarino, Claudio Masolo, and Guido Vetere. "OntoSeek: Content-based Access to the Web". *IEEE Inteligent Systems*, pages 70–80, May,June 1999.

[10] International Organization For Standardization. "Documentation - Guidelines for the establishment and development of monolingual thesauri", 1986. Ref. No ISO 2788-1986.

[11] Raili Kauppi. "Einfuhrung in die Theorie der Begriffssysteme". *Acta Universitatis Tamperensis, Ser A, Vol 15, University of Tampere*, 1967.

[12] Sean Luke, Lee Spector, David Rager, and Jim Hendler. "Ontology-based Web Agents". In *Proceedings of First International Conference on Autonomous Agents*, 1997. (http://www.cs.umd.edu/projects/plus/SHOE/).

[13] Zygmunt Mazur. "Models of a Distributed Information Retrieval System Based on Thesauri with Weights". *Information Processing and Management*, 30(1):61–77, 1994.

[14] Deborah L. McGuinness. "Ontological Issues for Knowledge-Enhanced Search". In *Proceedings of FOIS'98*, Trento, Italy, June 1998. Amsterdam, IOS Press.

[15] Carlo Meghini, Yannis Tzitzikas, and Nicola Spyratos. An abduction-based method for index relaxation in taxonomy-based sources. In *Proceedings of MFCS 2003, 28th International Symposium on Mathematical Foundations of Computer Science*, number 2747 in Lecture notes in computer science, pages 592–601, Bratislava, Slovak Republic, August 2003. Springer Verlag.

[16] C. Paice. "A Thesaural Model of Information Retrieval". *Information Processing and Management*, 27(5):433–447, 1991.

[17] Ruben Prieto-Diaz. "Implementing Faceted Classification for Software Reuse". *Communications of the ACM*, 34(5), 1991.

[18] S. R. Ranganathan. "The Colon Classification". In Susan Artandi, editor, *Vol IV of the Rutgers Series on Systems for the Intellectual Organization of Information*. New Brunswick, NJ: Graduate School of Library Science, Rutgers University, 1965.

[19] Giovanni M. Sacco. "Dynamic Taxonomies: A Model for Large Information Bases". *IEEE Transactions on Knowledge and Data Engineering*, 12(3), May 2000.

[20] Y. Tzitzikas, A. Analyti, N. Spyratos, and P. Constantopoulos. "An Algebra for Specifying Compound Terms for Faceted Taxonomies". In *13th European-Japanese Conf. on Information Modelling and Knowledge Bases*, Kitakyushu, J, June 2003.

[21] Y. Tzitzikas, N. Spyratos, and P. Constantopoulos. "Mediators over Ontology-based Information Sources". In *Second Int. Conf. on Web Information Systems Engineering, WISE 2001*, Kyoto, Japan, Dec. 2001.

[22] Yannis Tzitzikas and Carlo Meghini. "Query Evaluation in Peer-to-Peer Networks of Taxonomy-based Sources". In *Proceedings of 19th Int. Conf. on Cooperative Information Systems, CoopIS'2003*, Catania, Sicily, Italy, November 2003.

[23] Yannis Tzitzikas, Carlo Meghini, and Nicolas Spyratos. "Taxonomy-based Conceptual Modeling for Peer-to-Peer Networks". In *Proceedings of 22th Int. Conf. on Conceptual Modeling, ER'2003*, Chicago, Illinois, October 2003.

[24] Frank van Harmelen and Dieter Fensel. "Practical Knowledge Representation for the Web". In *Workshop on Intelligent Information Integration, IJCAI'99*, 1999.

[25] P. Zunde and M.E. Dexter. "Indexing Consistency and Quality". *American Documentation*, 20(3):259–267, July 1969.