Incremental Knowledge Acquisition for Non-Monotonic Reasoning

Fabrizio Sebastiani & Umberto Straccia
Istituto di Elaborazione dell'Informazione
Consiglio Nazionale delle Ricerche
Via S. Maria, 46 - 56126 Pisa (Italy)
E-mail : {fabrizio,straccia}@iei.pi.cnr.it
Phone : +39.50.593407
Fax : +39.50.554342

Abstract

The use of conventional non-monotonic reasoning tools in real-sized knowledge-based applications is hindered by the fact that the knowledge acquisition phase cannot be accomplished in the incremental way that is instead typical of knowledge base management systems based on monotonic logics. As a result, some researchers have departed from orthodox non-monotonic formalisms and proposed languages for the representation of *Multiple Inheritance Networks with Exceptions* (MINEs). Such languages do not suffer from the problem of incrementality in knowledge acquisition, but are inadequate both from a formal and from an empirical point of view. In fact, they are not endowed with a formal semantics, and the intuitions that underlie their inferential mechanisms are far from being widely agreed upon.

In this paper we discuss an approach to non-monotonic reasoning which does allow the phase of knowledge acquisition to be accomplished in an incremental and modular way, but at the same time relies on a solid and widely acknowledged formal apparatus such as

First Order Logic (FOL). We have obtained this by specifying a (non-monotonic) function that maps MINEs into sets of FOL formulae. We have shown that the mapping function we discuss is sound and complete, in the sense that each conclusion that can be derived from a MINE is also derivable from the set of FOL formulae resulting from its translation via the mapping function, and viceversa.

Keywords : Compositionality, Knowledge Representation, Knowledge Acquisition, Multiple Inheritance, Non-monotonic Reasoning

1 Introduction

Traditional formal systems have the property of being *monotonic*, in the sense that an addition to the set of formulae making up the knowledge base (KB) never determines a decrement in the set of conclusions that may be derived from the KB via the inference rules. Formally, if $\Gamma \vdash A$ holds, then necessarily $\Gamma \cup \Gamma' \vdash A$ holds, where Γ and Γ' are sets of formulae, A is a formula and $\vdash$ is the derivability relation.

Unfortunately, only few application domains exist whose formalization may be accomplished by means of monotonic formal systems[1]. In fact, there are many reasoning patterns for which a monotonic formal system would prove inadequate. One of these is *default reasoning*, a reasoning style that is to be applied whenever the rules involved allow for *exceptions*. Such rules dictate that, "by default" (i.e. whenever there is no information to the contrary), one may assume that the situation that is to be dealt with is a *typical* (i.e. not exceptional) situation. However, a possible subsequent acquisition of new knowledge might lead one to discover that this situation is not typical as was believed, but *exceptional*; in this case, what had been assumed "by default" is *revoked*. Hence, reasoning by default (also known as *reasoning in the presence of exceptions*) is a non-monotonic kind of reasoning.

An instance of this reasoning style is the inferential chain induced by the assertion "if x is a bird, assume that it flies, unless you know it does not". If we reason according to this rule and know that Opus is a bird, in the absence of information to the contrary we may conclude that Opus flies; but if we

[1]Witness the fact that most research papers about monotonic KR languages draw their examples from "unrealistically neat" application domains, such as e.g. geometry, kinships, and the like.

later learn that Opus is a penguin, knowing that penguins typically do not fly, we have to revoke the conclusion that Opus flies.

Non-monotonic reasoning is nowadays a hot topic in AI; in fact, the spectrum of its applications is extremely wide, ranging from image interpretation [8], reasoning about action [7], troubleshooting [10], knowledge bases automatic generation [12] and natural language understanding [1]. The requirements that are imposed by these application domains have lead to the development of a variety of non-monotonic formalisms, most of which belong to the offspring of Doyle and McDermott's Nonmonotonic Logic, Reiter's Default Logic and McCarthy's Circumscription[2]. Unfortunately, these formalisms, besides having unattractive metalogical and computational properties (i.e. undecidability of the versions with quantifiers, computational intractability of those without quantifiers), suffer from a problem that hinders their use in those knowledge representation contexts which require that the knowledge acquisition phase be accomplished in an incremental and modular fashion. We will call this problem the *Exceptions Explicitation Problem* (EEP).

Incrementality of knowledge acquisition is an asset of knowledge base management systems that hardly needs to be argued for. Large knowledge bases are the result of an evolutionary process; this happens because knowledge entry is a time-consuming process, and because knowledge may simply become available at later stages of the process, possibly contradicting (or specializing) previously acquired knowledge. When a large KB is built by this "stepwise refinement" process, it is highly desirable that the refinement consists in the plain, piecemeal *addition* of new knowledge chunks, rather than in a time-consuming *revision* (with a possibly ensuing deletion) of pre-existing chunks; in other words, it is desirable that knowledge acquisition be incremental (or *compositional*).

In Section 2 of this paper we will discuss how the EEP manifests itself in standard non-monotonic formalisms, and will see how this makes their implementation into working KBMSs allowing for incremental knowledge acquisition virtually impossible.

In Section 3 we will briefly describe a class of languages, those allowing the representation of *Multiple Inheritance Networks with Exceptions* (MINEs – see e.g. [13, 16]), that solve this problem by a technique we will call *implicit*

[2]Up-to-date accounts of what is going on in the field may be gathered by consulting [17, 3, 9].

handling of exceptions. Unfortunately, such languages lack a denotational semantics that account for the meaning of the linguistic primitives contained therein in a clear and unambiguous way. Besides, given the complexity and awkwardness of the non-standard proof theory such languages rely on in order to specify the conclusions that are derivable from a KB, it is substantially implausible to think that such a proof theory might "implicitly" constitute a semantics for the languages in question[3].

In order to overcome these deficiencies, in Section 4 we will propose an approach to non-monotonic reasoning that combines the advantages (in terms of incrementality of the knowledge acquisition phase) that are offered by the languages for MINEs, with those (in terms of semantic clarity) that are offered by a formally solid apparatus such as First Order Logic (FOL). This approach is accomplished by specifying a (non-monotonic) function that maps a MINE into a set of FOL formulae. We will show that this mapping function is a "good" mapping, in the sense that each conclusion which is derivable from a MINE is also derivable from the set of FOL formulae that results from the application of the mapping function to the MINE, and viceversa. An interesting side-effect of this result is that MINE-like reasoning can be performed by means of ordinary, ready-made first order theorem provers, with no need to add specific machinery. Section 5 concludes.

2 The Exceptions Explicitation Problem

In this section we will discuss, by means of a concrete example, how the EEP manifests itself in the context of Nonmonotonic Logic (NML); to this respect, other formalisms such as Default Logic and Circumscription behave in a completely analogous way, and will not be discussed. The "penguin" example that we have hinted at in the preceding section may be represented by means of the NML formula

$$\forall x \; Bird(x) \wedge M[Flies(x)] \Rightarrow Flies(x) \tag{1}$$

(where $M[\alpha]$ informally means "α does not contradict the KB"). Let us

[3]For a more detailed discussion on this "implicit" approach to the semantics of representation languages see e.g. [2].

consider an exception to the rule, "penguins are birds that typically do not fly", and let us represent it by means of the axioms:

$$\forall x\ Penguin(x) \Rightarrow Bird(x) \tag{2}$$
$$\forall x\ Penguin(x) \wedge M[\neg Flies(x)] \Rightarrow \neg Flies(x) \tag{3}$$

There is a problem hidden in this set of formulae: in NML the addition of the assertion $Penguin(opus)$ to axioms (1)-(3) generates two different "sets of conclusions" ("fixpoints", in NML terminology): in one of them Opus flies while in the other Opus does not fly, depending on whether we "first" consider axiom (1) or axiom (3), respectively; this happens regardless of the fact that our knowledge of the animal world makes us strongly prefer the conclusion according to which Opus does not fly.

Hence NML seems, at first sight, inadequate to account for this kind of reasoning, as it generates a situation of ambiguity in the representation of a state of affairs that is all but ambiguous to us[4]. At first sight, it seems possible to overcome this problem without departing from NML; what one needs to do is *explicitly* consider exceptions in each default formula they involve. Accordingly, a set of axioms that induces a correct behaviour of NML for our example is the following:

$$\forall x\ Bird(x) \wedge M[Flies(x) \wedge \neg Penguin(x)] \Rightarrow Flies(x) \tag{4}$$
$$\forall x\ Penguin(x) \Rightarrow Bird(x)$$
$$\forall x\ Penguin(x) \wedge M[\neg Flies(x)] \Rightarrow \neg Flies(x)$$

At this point, given $Penguin(opus)$, axiom (4) does not allow one to conclude $Flies(opus)$; this leaves us with one fixpoint only, a fixpoint that contains only the conclusions we actually subscribe to.

However, this solution looks like an *ad hoc* patch rather than a real solution. In fact, it should be noted that, given the informal meaning that Doyle and McDermott attribute to NML formulae, the conclusion according to which Opus does not fly should already follow from axioms (1)-(3), as

[4]The generation of multiple fixpoints in NML may otherwise be due to an *inherent* ambiguity of the state of affairs being represented. A very famous example of this is the so-called "Nixon diamond" (see Example 2).

these already represent the fact that penguins are exceptional birds as far as the ability to fly is involved; consequently, the modification of axiom (1) to yield (4) seems a redundant re-assertion of this fact.

Besides, one should note that, until the KB did not contain any reference to penguins, axiom (1) was more than adequate. The introduction of axioms (2) and (3) has obliged us to transform axiom (1) into axiom (4); in the more general case, the acquisition of new knowledge obliges us to *revoke* (in order to subsequently assert some modified version of them) *previously acquired formulae from the KB*. It is then evident that the realization of default reasoning by means of NML is undermined by the impossibility to update KBs by simple piecemeal additions of new knowledge chunks. In semantic terms this means the impossibility, unlike what happens with traditional logical formalisms, to characterize the extensional semantics of the modified KB in a compositional way, i.e. as a combination of the extensions of the original KB and of the newly acquired formulae.

But the need to explicitly represent exceptions brings about other, even more compelling problems, having to do with the way in which, at the time of the introduction of a new formula into the KB, it is to be determined which are the formulae that are to be modified and which are the ones that are not. It turns out that this operation cannot be realized by simply performing a number of matchings between the formula to be introduced and the other formulae in the KB; on the contrary, it requires in general *a NML theorem proving operation*. This may clearly be seen by taking a look at our example: the relation between the precondition of axiom (1) and the precondition of axiom (3) has been determined by relying on the fact that $Bird(x)$ is *derivable* from $Penguin(x)$ by means of axiom (2); the transformation of axiom (1) into axiom (4) is then realized in order to "inhibit" not axiom (1) in itself, but the inferential chain that would lead us to conclude $Flies(x)$ from $Penguin(x)$. In general, given a NML knowledge base, it is not even determined *a priori* whether there exist one, several or no inferential chains that lead to the undesired conclusion (in our case, one, several or no inferential chains between $Penguin(x)$ and $Flies(x)$); in case several such chains exist, *each of them* has to be inhibited by means of a call to the NML theorem prover, each resulting in the introduction of a subformula of type $M[\alpha]$.

That the phase of KB construction requires repeated calls to the NML theorem prover in order to maintain the consistency of the KB that is being built, is in itself rather implausible; but the whole endeavour becomes absurd

once we consider that NML is undecidable! This means that, unless the construction of the KB is realized in a completely *static* (non incremental) way, the problem of knowledge acquisition in NML (and, analogously, in the other non-monotonic formalisms mentioned in Section 1) is an *unsolvable* problem.

3 Multiple Inheritance Networks with Exceptions

The Exceptions Explicitation Problem that affects standard non-monotonic formalisms has lead to the definition of languages for the representation of *Multiple Inheritance Networks with Exceptions* (MINEs). Such languages have two fundamental characteristics:

- they are specifically oriented to the representation of (non-monotonic) knowledge of a *taxonomic* nature (i.e. of knowledge that lends itself to being organized hierarchically); hence, they are less expressive of those we have mentioned in Section 1, in the sense that they only allow for monadic predicate symbols, a limited use of negation, and no disjunction at all;
- they implement an *implicit* handling of exceptions (see below).

A MINE is a directed graph whose nodes represent classes or individuals of the domain of discourse, and whose edges represent relationships of "conceptual containment" between the nodes involved. For instance, a MINE Γ might contain the nodes *opus*, $Penguin$ and $Bird$, and contain the edges $opus \rightarrow Penguin$, $Bird \rightarrow Flies$ and $Penguin \not\rightarrow Flies$. The word *inheritance* refers to the fact that the deductive mechanisms of these languages are such that an edge $Bird \rightarrow Flies$ has the effect of transmitting "by inheritance" all the properties of $Flies$ to $Bird$; this inheritance is *multiple* because a MINE Γ may contain more than one edge outgoing from the same node, the net effect being that the node inherits properties from different, unrelated nodes. Last, we speak of networks *with exceptions* because the intended denotation of an edge such as $p \rightarrow q$ is the set of states of affairs in which "p's are typically q's"; this assertion allows for the existence of p's

that are not q's, i.e. of exceptions. Dual arguments apply to edges of type $p \not\rightarrow q$.

The languages for the representation of MINEs that have been proposed in the literature are by now a fairly vast and multifaceted class (see e.g. [4, 5, 11, 15]). It is not our purpose to give an exhaustive review of the constructs and inferential mechanisms that appear in the various languages (to this respect, the interested reader may consult [13, 16]); what we want is to introduce a number of fundamental concepts that, besides being common to a vast class of MINE representation languages, will be sufficient to illustrate the problems affecting such languages in general, and to motivate the approach described in Section 4.

In the discussion that follows we will restrict ourselves to the case in which the relation $\rightarrow \cup \not\rightarrow$ (a relation which we will indicate with the symbol $\rightsquigarrow$) is irreflexive and transitive (i.e. a *strict partial order*); this limitation is not particularly stringent from a pragmatic point of view. In MINEs that enjoy this property, exceptions are *implicitly* handled by this ordering: to a first approximation, we can say that, in case of conflicts, an edge is "preferred" to another if the source node of the first precedes the source node of the second in the ordering. For example, given the MINE consisting of the edges $opus \rightarrow Penguin$, $Penguin \rightarrow Bird$, $Bird \rightarrow Flies$, $Penguin \not\rightarrow Flies$, we will see how the conclusion according to which Opus does not fly is given priority, on the grounds that the premise of $Penguin \not\rightarrow Flies$ is more specific than that of $Bird \rightarrow Flies$, and that, as a consequence, the conclusion derivable from the former is more reliable than the one derivable from the latter[5].

Let us now formally define our language for the representation of MINEs.

Definition 1 *Let $\mathcal{I}$ and $\mathcal{C}$ be two finite sets of* individual constants *and* predicate symbols*, respectively, and let $\mathcal{A}$ be a finite set of* positive edges $x \rightarrow y$ *and* negative edges $x \not\rightarrow y$*, where $x \in \mathcal{I} \cup \mathcal{C}$ and $y \in \mathcal{C}$. A* Multiple Inheritance Network with Exceptions *(MINE) is a triple $\Gamma = \langle \mathcal{I}, \mathcal{C}, \mathcal{A} \rangle$ such*

[5]The reader may have noticed that in this example we have used the edge $Penguin \rightarrow Bird$ to formalize a fact that would be better represented by an implication of classical logic, given that *all* penguins are birds. The integration between "categorical" and "default" implications in a single framework is not among the purposes of this paper; for simplicity, we will hereafter take all implications as having a default nature. For a more detailed discussion on the above-mentioned integration see e.g. [4].

that the relation $\rightarrow \cup \not\rightarrow$ *is a strict partial order. The elements of the set* $\mathcal{I} \cup \mathcal{C}$ *are called* nodes. *The* polarity *of an edge is given by its positive or negative sign.*

With Γ_I, Γ_C and Γ_A we will refer to the components $\mathcal{I}$, $\mathcal{C}$ and $\mathcal{A}$ of Γ. In the examples that follow we will use lowercased words as metavariables ranging over individual constants, capitalized words for predicate symbols and the letters $p, q, \ldots, x, y, z$ for (generic) nodes.

Let us now define, given a MINE Γ, what the set of conclusions that may be derived from Γ is; this is very important because, similarly to what happens in other logics for knowledge representation, it establishes the way in which we may infer knowledge *implicit* in the KB from the knowledge explicitly coded therein. However, in order to do so we have to introduce the notions of "chain" and "path".

Definition 2 *A* chain *in a MINE* Γ *is a sequence of edges in* Γ_A*,* $x_1 \rightsquigarrow x_2 \rightsquigarrow \ldots \rightsquigarrow x_{n-1} \rightsquigarrow x_n$ *(*$n \geq 2$*). The node* x_1 *is called* initial node *of the chain and* x_n *is called* terminal node*. The* degree *of a chain in* Γ *with initial node* x *and terminal node* y *is given by the length of the longest chain from* x *to* y *in* Γ*.*

Hereafter we will use lowercased greek letters as metavariables ranging over chains, and we will indicate with $deg_\Gamma(\sigma)$ the degree of a chain σ on Γ.

Definition 3 *A* positive path *in a MINE* Γ *is a chain* $x_1 \rightarrow x_2 \rightarrow \ldots \rightarrow x_{n-1} \rightarrow x_n$ *(*$n \geq 2$*) in* Γ *consisting of positive edges only. A* negative path *in a MINE* Γ *is a chain* $x_1 \rightarrow x_2 \rightarrow \ldots \rightarrow x_{n-1} \not\rightarrow x_n$ *(*$n \geq 2$*) in* Γ *consisting of a (possibly empty) sequence of positive edges followed by one negative edge. The* polarity *of a path is given by the polarity of its last edge. The edge formed by the initial node of a path, its terminal node and its polarity is called* conclusion *derived from the path.*

Intuitively, paths are those chains from which it is somehow possible to draw conclusions. For example, let us consider a MINE such as $KB1 = \langle \{opus\}, \{Penguin, Bird, Flies, Swims\}, \{opus \rightarrow Penguin, Penguin \rightarrow Bird, Bird \rightarrow Flies, Penguin \not\rightarrow Flies, Flies \not\rightarrow Swims\} \rangle$. The chain

$$opus \rightarrow Penguin \rightarrow Bird \rightarrow Flies \tag{5}$$

is a positive path, from which we can draw the conclusion $opus \rightarrow Flies$ ("As there is no information to the contrary, we assume that Opus flies"), while from the negative path

$$opus \rightarrow Penguin \not\rightarrow Flies \tag{6}$$

we may conclude $opus \not\rightarrow Flies$ ("As there is no information to the contrary, we assume that Opus does not fly"). The chain

$$opus \rightarrow Penguin \not\rightarrow Flies \not\rightarrow Swims \tag{7}$$

is not a path; in fact, no reasonable conclusion may be drawn from it with respect to the fact whether Opus can swim or not. Similar empirical considerations have induced researchers to define the notion of "conclusion" with respect to paths only, and not with respect to chains in general.

As we have seen, MINEs such as $KB1$ have paths from which contradictory conclusions may be derived; unfortunately, this generates situations of ambiguity in the representation of states of affairs which are all but ambiguous to us. In order to eliminate these situations, what we will do is to define which are, given a MINE Γ, the paths from which we may derive "reliable" conclusions, and that must consequently be given priority over others. For example, in a MINE such as $KB1$ this will allow us to give path (6) priority over path (5), hence to inhibit the undesired conclusion according to which Opus flies. In MINE terminology, paths from which we may derive "reliable" conclusions are called *derivable* paths.

Definition 4 *A positive path σ is* derivable *from a MINE Γ (in symbols: $\Gamma \vdash \sigma$) if and only if the following conditions obtain:*

1. *if $\sigma = x \rightarrow y$, then $\Gamma \vdash \sigma$ if and only if $\sigma \in \Gamma_A$ and $x \not\rightarrow y \notin \Gamma_A$;*
2. *if $deg_\Gamma(\sigma) = n > 1$, then $\Gamma \vdash \sigma$ if and only if there exists a (possibly empty) positive path δ and nodes x, y and $u \in \Gamma_I \cup \Gamma_C$ such that:*
   (a) $\sigma = x \rightarrow \delta \rightarrow u \rightarrow y$
   (b) $u \rightarrow y \in \Gamma_A$;

(c) $\Gamma \vdash x \rightarrow \delta \rightarrow u$;

(d) $x \not\rightarrow y \notin \Gamma_A$;

(e) *for all* v *and for all* τ *such that* $\Gamma \vdash x \rightarrow \tau \rightarrow v$ *and* $v \not\rightarrow y \in \Gamma_A$, *there exists a* z *such that* $z \rightarrow y \in \Gamma_A$ *and either* $z = x$ *or* $\Gamma \vdash x \rightarrow \gamma_1 \rightarrow z \rightarrow \gamma_2 \rightarrow v$ *for some path* γ_1 *and for some path* γ_2.

Definition 5 *A negative path* σ *is* derivable *from a MINE* Γ *(in symbols:* $\Gamma \vdash \sigma$*) if and only if the following conditions obtain:*

1. *if* $\sigma = x \not\rightarrow y$, *then* $\Gamma \vdash \sigma$ *if and only if* $\sigma \in \Gamma_A$ *and* $x \rightarrow y \notin \Gamma_A$

2. *if and only if* $deg_\Gamma(\sigma) = n > 1$, *then* $\Gamma \vdash \sigma$ *if and only if there exists a (possibly empty) positive path* δ *and nodes* x, y *and* $u \in \Gamma_I \cup \Gamma_C$ *such that:*

   (a) $\sigma = x \rightarrow \delta \rightarrow u \not\rightarrow y$

   (b) $u \not\rightarrow y \in \Gamma_A$

   (c) $\Gamma \vdash x \rightarrow \delta \rightarrow u$

   (d) $x \rightarrow y \notin \Gamma_A$

   (e) *for all* v *and for all* τ *such that* $\Gamma \vdash x \rightarrow \tau \rightarrow v$ *and* $v \rightarrow y \in \Gamma_A$ *there exists a* z *such that* $z \not\rightarrow y \in \Gamma_A$ *and either* $z = x$ *or* $\Gamma \vdash x \rightarrow \gamma_1 \rightarrow z \rightarrow \gamma_2 \rightarrow v$ *for some path* γ_1 *and for some path* γ_2.

Following the terminology of [16] we might say that clauses (b) and (c) of Definitions 4 and 5 represent the option for a style of reasoning informed by "bottom-up chaining", while clause (e) represents the option for "off-path preclusion"; the mapping function that we will describe in Section 4 subscribes to these options[6].

[6]The mapping function may be modified without difficulty in case one wants to opt for top-down chaining and/or on-path preclusion. Our notion of derivability is very similar to the one proposed in [5]; it differs from it in the fact that in [5] item (1) of the two definitions specifies that, if $deg_\Gamma(\sigma) = 1$, then $\Gamma \vdash \sigma$ if and only if $\sigma \in \Gamma_A$ (i.e. unlike in our definition, it is not necessary that the edge with opposite polarity does not belong to Γ_A). The modifications that could be made to our formalism in order to obtain the behaviour described in [5] would be trivial, and will not be considered.

Definition 6 *The* set of conclusions *of a MINE* Γ *(in symbols:* $C(\Gamma)$*) is the set of those edges* $p \rightsquigarrow q$ *such that* $p \rightsquigarrow q$ *is the conclusion of a path* σ *derivable from* Γ*.*

The set of conclusions that may be drawn from a MINE Γ is the set that represents the knowledge "implicitly" present in the KB.

4 Mapping MINEs into FOL

The implicit handling of exceptions implemented in MINE representation languages by means of the ordering of nodes in the network allows the phase of knowledge acquisition to be realized in an incremental way: recalling the example of Section 2, if our KB contained the edge $Bird \rightarrow Flies$, a subsequent introduction of the edges $opus \rightarrow Penguin$, $Penguin \rightarrow Bird$ and $Penguin \not\rightarrow Flies$ would not bring about the need to modify the preexisting edge, as the new strict partial order deriving from the introduction of the new edges would inhibit the undesired conclusion $opus \rightarrow Flies$.

The limit of the MINE-based approach is its lack of semantic clarity. In fact, the development of a model-theoretic semantics for MINE representation languages is still an open problem, and the lack of such a semantics makes both the analysis of these languages and the comparison between them extremely difficult.

In this paper we discuss an approach to non-monotonic reasoning that allows for incremental knowledge acquisition, while at the same time relying on a formally solid and widely acknowledged framework such as First Order Logic. We have accomplished this by specifying a first order representation of the MINE formalism (a representation that we dub LT, the *Logical Theory of Multiple Inheritance with Exceptions*), and a mapping function $\mathcal{M}$ that maps a MINE Γ into a set of FOL formulae. The LT theory will be on all counts our formalism for default reasoning; a *mathematical* theory for default reasoning (i.e. a KB) will be obtained by adding to LT the result of the application of $\mathcal{M}$ to a particular MINE, obtaining the FOL knowledge base $\mathcal{T}(\Gamma) = \mathcal{M}(\Gamma) \cup LT$. The net effect is that the LT component of $\mathcal{T}(\Gamma)$ remains fixed throughout the phase of knowledge acquisition (in the same sense in which the logical axioms are fixed for a given logical system), while the only component that varies is the $\mathcal{M}(\Gamma)$ component.

It may be shown that $\mathcal{T}$ is sound and complete, in the sense that each conclusion derivable from a MINE Γ is also derivable from $\mathcal{T}(\Gamma)$, and viceversa.

At this point a question arises naturally: how can a non-monotonic system be mapped into FOL, a notoriously monotonic system, while at the same time maintaining non-monotonicity? Isn't this a contradiction in terms? In order to hint that this is a plausible state of affairs, let us recall that Circumscription (one of the most credited non-monotonic formalisms) is nothing else but a mapping function from theories of a formalism F_1 (FOL) into theories of another formalism F_2 (second order logic), and that in this case too the target formalism is monotonic. In our case (and in the case of Circumscription too), non-monotonicity is a property *of the mapping function*, and not of the target formalism. In fact, if Γ and Γ' are MINEs such that $\Gamma \cup \Gamma'$ is the MINE obtained through the union of the respective components (that is: $\Gamma \cup \Gamma' = \langle \Gamma_I \cup \Gamma'_I, \Gamma_C \cup \Gamma'_C, \Gamma_A \cup \Gamma'_A \rangle$), $\mathcal{M}$ is such that the translation of Γ is not in general a logical consequence of the translation of $\Gamma \cup \Gamma'$; that is, in general $\mathcal{M}(\Gamma \cup \Gamma') \not\models \mathcal{M}(\Gamma)$. This is a necessary condition for any function aspiring to translate a non-monotonic formal system into a monotonic one; if $\mathcal{S}_1$ is a non-monotonic system based on language $\mathcal{L}$, E and E' are sets of expressions on $\mathcal{L}$, and $\mathcal{K}$ is the translation function of sets of expressions of $\mathcal{L}$ into sets of expressions of a monotonic system $\mathcal{S}_2$, then $\mathcal{K}$ must be such that in general it is not the case that $\mathcal{K}[E \cup E'] \models_{S_2} \mathcal{K}[E]$.

In order to prove our result, we will proceed in the following way. First of all, we will define the mapping function $\mathcal{M}$; we will then define LT, the Logical Theory of Multiple Inheritance with Exceptions, and will then show that belonging to the set of conclusions of a MINE Γ is equivalent to being a logical consequence in LT (a notion we will indicate with the symbol "$\models_{LT}$") of $\mathcal{M}\Gamma$[7]. All this will be described in a semi-informal way in Sections 4.1 and 4.2, and subsequently formalized in Section 4.3.

4.1 The $\mathcal{M}$ function

The purpose of the $\mathcal{M}$ function is that of translating a MINE $\Gamma = \langle \mathcal{I}, \mathcal{C}, \mathcal{A} \rangle$ into a set of formulae $\mathcal{M}(\Gamma)$ constituting a first order representation of the

[7]Let us recall that in the language of mathematical logic, $\alpha \models_A \beta$ is short for $A \cup \{\alpha\} \models \beta$, where α and β are formulae and A is a theory of the language in question.

graph formed by the nodes and edges of Γ. In order to obtain this, first of all let us represent every node p in $\Gamma_I \cup \Gamma_C$ by an individual constant p of the first order language. The edges of Γ_A are instead represented through instances of the binary predicates $ISDA$ ("IS Directly A") and $ISDNA$ ("IS Directly Not A"): the formulae $ISDA(p,q)$ and $ISDNA(p,q)$ represent then the fact that edges $p \rightarrow q$ and $p \not\rightarrow q$, respectively, belong to Γ_A. Let us notice that, in order for Γ_A to be represented adequately, the following conditions must obtain:

- the predicate $ISDA$ must be such that $p \rightarrow q \in \Gamma_A$ if and only if every model of $\mathcal{M}(\Gamma)$ is also a model of $ISDA(p,q)$;
- the predicate $ISDNA$ must be such that $p \not\rightarrow q \in \Gamma_A$ if and only if every model of $\mathcal{M}(\Gamma)$ is also a model of $ISDNA(p,q)$.

In order to accomplish this, let us consider for every edge $p \rightarrow q \in \Gamma_A$ a formula of type $ISDA(p,q)$. It will be apparent in the proof of Lemma 1 that we need these formulae to be all and *the only* instances of the $ISDA$ predicate to be true in the models of $\mathcal{M}(\Gamma)$; in order to accomplish this, we impose that the following axiom holds[8]:

$$\forall x \forall y \; ISDA(x,y) \Leftrightarrow \bigvee_{p \rightarrow q \in \Gamma_A} (x = p \wedge y = q) \tag{8}$$

Analogously, if we want to represent negative edges by means of the predicate $ISDNA$, we obtain the formula

$$\forall x \forall y \; ISDNA(x,y) \Leftrightarrow \bigvee_{p \not\rightarrow q \in \Gamma_A} (x = p \wedge y = q) \tag{9}$$

In order to obtain a one-to-one correspondence between the network and its FOL representation, we have also to consider that, in any MINE, nodes are considered as symbols referring to individuals "implicitly" different from each other; this is often an unstated assumption in such languages, but it is

[8]This axiom turns out to be equivalent [6] to the *minimization* of the predicate $ISDA$ by means of the Circumscription axiom $Circum(\bigwedge_{p \rightarrow q \in \Gamma_A} ISDA(p,q); ISDA; ())$; alternatively, this minimization may be seen as the *completion* of the $ISDA$ predicate.

nevertheless "wired" in the reasoning machinery that implements them. As a consequence, also individual constants must be interpreted as referring each to a different individual: that is, we do not want that models of $\mathcal{M}(\Gamma)$ may exist that associate syntactically different constants to the same individual of the domain of discourse. In order for this to hold, we impose that the axiom

$$\bigwedge_{\{p,q\}\in\Gamma_I\cup\Gamma_C}(p \neq q) \tag{10}$$

holds, an axiom that formalizes what is usually known as the "unique names assumption".

This concludes the definition of the $\mathcal{M}$ function, a function that may then be concisely described by means of the following definition.

Definition 7 *Let Γ be a MINE; $\mathcal{M}$ is the function that maps Γ into the FOL axiom*

$$\begin{array}{l}(\forall x \forall y\ ISDA(x,y) \Leftrightarrow \bigvee_{p\rightarrow q\in\Gamma_A}(x=p\wedge y=q))\wedge\\ (\forall x \forall y\ ISDNA(x,y) \Leftrightarrow \bigvee_{p\not\rightarrow q\in\Gamma_A}(x=p\wedge y=q))\wedge\\ (\bigwedge_{\{p,q\}\in\Gamma_I\cup\Gamma_C}(p\neq q))\end{array}$$

Note that the size of the axiom is at most $O(n^2)$, where n is the number of nodes in Γ.

At this point we may already discuss how incrementality is achieved in our framework. In the preceding section we have hinted at the fact that the LT component of $\mathcal{T}(\Gamma)$ remains fixed throughout the phase of knowledge acquisition, while the only component that varies is the $\mathcal{M}(\Gamma)$ component. We may now see that the way in which this component varies is informed by the principles of incrementality and compositionality. In fact, if one needs to add the equivalent of an edge $r \rightarrow s$ (resp. $r \not\rightarrow s$) to a knowledge base $\mathcal{T}(\Gamma)$, one needs only to add the formula $(x = r \wedge y = s)$ as a further operand of the disjunction $\bigvee_{p\rightarrow q\in\Gamma_A}(x = p \wedge y = q)$ (resp. $\bigvee_{p\not\rightarrow q\in\Gamma_A}(x = p \wedge y = q)$), and the formula $(r \neq s)$ to the conjunction $\bigwedge_{\{p,q\}\in\Gamma_I\cup\Gamma_C}(p \neq q)$. As required, knowledge acquisition becomes a matter of simple piecemeal additions of new chunks of knowledge to the preexisting KB.

4.2 The Logical Theory of Multiple Inheritance with Exceptions

Now that we have defined the $\mathcal{M}$ function, let us build a first order theory LT such that a formula of type $ISA(x, y)$ (resp. $ISNA(x, y)$) is derivable from $\mathcal{M}(\Gamma)$ in LT if and only if $x \rightarrow y$ (resp. $x \not\rightarrow y$) belongs to $C(\Gamma)$, the set of conclusions of Γ. The LT theory will roughly consist of the definitions of the predicates ISA and $ISNA$ in terms of the predicates $ISDA$ and $ISDNA$ we have seen in the preceding section. This definition is rather complex; hence, it will be introduced in an incremental way, stepping through the definition of some other auxiliary predicates representing the existence of particular conditions on the topology of Γ.

In order to do so, let us first start by defining two predicates $\Pi_P(x, t, y)$ and $\Pi_N(x, t, y)$ representing the derivability in Γ of a path (positive or negative, respectively) with initial node x and terminal node y passing from node t. As a consequence, the following conditions must obtain:

- the predicate Π_P must be such that $\Gamma \vdash x \rightarrow \gamma_1 \rightarrow t \rightarrow \gamma_2 \rightarrow y$, for some γ_1 and γ_2, if and only if every model of $\mathcal{M}(\Gamma) \cup LT$ is also a model of $\Pi_P(x, t, y)$;
- the predicate Π_N must be such that $\Gamma \vdash x \rightarrow \gamma_1 \rightarrow t \rightarrow \gamma_2 \not\rightarrow y$, for some γ_1 and γ_2, if and only if every model of $\mathcal{M}(\Gamma) \cup LT$ is also a model of $\Pi_N(x, t, y)$.

In order for Π_P and Π_N to have the desired characteristics, let us define them by means of a "literal" translation of Definitions 4 and 5. The translation of Definition 4 is obtained by imposing that the following formula (which has been subdivided in several numbered subformulae and indented so as to highlight its structural affinity with the definition) belongs to LT.

$$
\begin{array}{lll}
\forall x \forall t \forall y\ \Pi_P(x,t,y) \Leftrightarrow & & \\
\quad ((x = t \wedge ISDA(x,y) \wedge \neg ISDNA(x,y)) & & 1. \\
\vee \quad (((\exists t' \Pi_P(x,t,t') \wedge t \neq t' \wedge ISDA(t',y)) & & 2.(a)-(b) \\
\vee \quad (\exists t' \Pi_P(x,t',t) \wedge t \neq t' \wedge ISDA(t,y))) \wedge & & \\
\quad \neg ISDNA(x,y) & & (d) \\
\wedge \quad (\forall v \exists v' (\Pi_P(x,v',v) \wedge ISDNA(v,y)) & & (e) \\
\quad \Rightarrow (\exists z ISDA(z,y) \wedge (z = x \vee \Pi_P(x,z,v)))))) & &
\end{array}
$$

At this point we may characterize the conclusion that can be drawn from a positive path, by means of a predicate $ISA(x, y)$, by adding to LT the simple formula

$$\forall x \forall y \ (ISA(x, y) \Leftrightarrow \exists t \ \Pi_P(x, t, y))$$

This mirrors the fact that, in any MINE Γ, $x \rightarrow y$ may be inferred if and only if a positive path is derivable that has x as initial node and y as terminal node. The predicate Π_N is defined in a completely analogous way to Π_P.

This concludes the definition of LT, a FOL theory that may then be concisely described by means of the following definition.

Definition 8 *The* Logical Theory of Multiple Inheritance with Exceptions *(LT) is the FOL theory* $LT \equiv \{A_1, A_2, A_3, A_4\}$*, where:*

- $A_1 \equiv \ \forall x \forall t \forall y \ \Pi_P(x, t, y) \Leftrightarrow$
  $((x = t \wedge ISDA(x, y) \wedge \neg ISDNA(x, y))$
  $\vee \ (((\exists t' \Pi_P(x, t, t') \wedge t \neq t' \wedge ISDA(t', y)) \vee$
  $(\exists t' \Pi_P(x, t', t) \wedge t \neq t' \wedge ISDA(t, y))) \wedge$
  $\neg ISDNA(x, y) \wedge$
  $(\forall v \exists v' (\Pi_P(x, v', v) \wedge ISDNA(v, y))$
  $\Rightarrow (\exists z ISDA(z, y) \wedge (z = x \vee \Pi_P(x, z, v))))))$

- $A_2 \equiv \ \forall x \forall t \forall y \ \Pi_N(x, t, y) \Leftrightarrow$
  $((x = t \wedge ISDNA(x, y) \wedge \neg ISDA(x, y))$
  $\vee \ (((\exists t' \Pi_P(x, t, t') \wedge t \neq t' \wedge ISDNA(t', y)) \vee$
  $(\exists t' \Pi_P(x, t', t) \wedge t \neq t' \wedge ISDNA(t, y))) \wedge$
  $\neg ISDA(x, y) \wedge$
  $(\forall v \exists v' (\Pi_P(x, v', v) \wedge ISDA(v, y))$
  $\Rightarrow (\exists z ISDNA(z, y) \wedge (z = x \vee \Pi_P(x, z, v))))))$

- $A_3 \equiv \forall x \forall y \ (ISA(x, y) \Leftrightarrow \exists t \ \Pi_P(x, t, y))$*;*

- $A_4 \equiv \forall x \forall y \ (ISNA(x, y) \Leftrightarrow \exists t \ \Pi_N(x, t, y))$

4.3 An equivalence result

So far we have described $\mathcal{M}(\Gamma)$ and LT, the two components of a (non-monotonic) function that maps MINEs into sets of FOL formulae. At this

point the only thing we have to do is to describe the properties of $\mathcal{T}(\Gamma) \equiv \mathcal{M}(\Gamma) \cup LT$. By exploiting the acylicity of our MINEs, we are able to prove the following fundamental lemma by finite induction on the degree of paths.

Lemma 1 (Correspondence) *Let Γ be a MINE and $\mathbf{M}$ be any first order interpretation such that $\mathbf{M} \models_{LT} \mathcal{M}(\Gamma)$. Then, for all $\langle x, y\rangle \subseteq (\Gamma_I \cup \Gamma_C) \times (\Gamma_I \cup \Gamma_C)$:*

1. $x \rightarrow y \in C(\Gamma) \Longleftrightarrow \mathbf{M} \models_{LT} ISA(x, y)$
2. $x \not\rightarrow y \in C(\Gamma) \Longleftrightarrow \mathbf{M} \models_{LT} ISNA(x, y)$ ∎

The proofs of Lemma 1 and of the lemmas that follow are given in the appendix. The following lemma derives immediately from Lemma 1.

Lemma 2 (Completion) *Let Γ be a MINE and $\mathbf{M}$ be any first order interpretation such that $\mathbf{M} \models_{LT} \mathcal{M}(\Gamma)$. Then, for all $\langle x, y\rangle \subseteq (\Gamma_I \cup \Gamma_C) \times (\Gamma_I \cup \Gamma_C)$:*

1. $x \rightarrow y \notin C(\Gamma) \Longleftrightarrow \mathbf{M} \models_{LT} \neg ISA(x, y)$
2. $x \not\rightarrow y \notin C(\Gamma) \Longleftrightarrow \mathbf{M} \models_{LT} \neg ISNA(x, y)$ ∎

By combining Lemma 1, Lemma 2 and the Consistency Lemma for "$\vdash$" proven in [5] we obtain

Lemma 3 (Consistency) *Let Γ be a MINE and let $\mathbf{M}$ be any first order interpretation such that $\mathbf{M} \models_{LT} \mathcal{M}(\Gamma)$. Then, for all $\langle x, y\rangle \subseteq (\Gamma_I \cup \Gamma_C) \times (\Gamma_I \cup \Gamma_C)$:*

1. $\mathbf{M} \models_{LT} ISA(x, y) \Longrightarrow \mathbf{M} \models_{LT} \neg ISNA(x, y)$
2. $\mathbf{M} \models_{LT} ISNA(x, y) \Longrightarrow \mathbf{M} \models_{LT} \neg ISA(x, y)$ ∎

Hence, this lemma shows the correct relation that exists between ISA and $ISNA$.

The results dealt with in the preceding lemmas presuppose the existence of at least one model of $\mathcal{M}(\Gamma)$ with respect to LT. We can prove that, in fact, such a model always exists. The following lemma is proven in a constructive way, i.e. by deriving the required model from Γ itself.

Lemma 4 *Let* Γ *be a MINE. Then there exists a first order interpretation* $\mathbf{M}$ *such that* $\mathbf{M} \models_{LT} \mathcal{M}(\Gamma)$. ∎

Finally, we may summarize the results of Lemmas 1-4 into the following theorem that completely characterizes $\mathcal{M}(\Gamma) \models_{LT}$.

Theorem 1 *Let* Γ *be a MINE. Then the following propositions hold*

1. $x \rightarrow y \in C(\Gamma) \Longleftrightarrow \mathcal{M}(\Gamma) \models_{LT} ISA(x, y)$
2. $x \not\rightarrow y \in C(\Gamma) \Longleftrightarrow \mathcal{M}(\Gamma) \models_{LT} ISNA(x, y)$
3. $x \rightarrow y \notin C(\Gamma) \Longleftrightarrow \mathcal{M}(\Gamma) \models_{LT} \neg ISA(x, y)$
4. $x \not\rightarrow y \notin C(\Gamma) \Longleftrightarrow \mathcal{M}(\Gamma) \models_{LT} \neg ISNA(x, y)$
5. $\mathcal{M}(\Gamma) \models_{LT} ISA(x, y) \Longrightarrow \mathcal{M}(\Gamma) \models_{LT} \neg ISNA(x, y)$
6. $\mathcal{M}(\Gamma) \models_{LT} ISNA(x, y) \Longrightarrow \mathcal{M}(\Gamma) \models_{LT} \neg ISA(x, y)$ ∎

Clauses (1) and (2) of Theorem 1 show that $\mathcal{T}$ is a *complete* translation of MINEs into FOL, in the sense that, for each conclusion that can be derived from a MINE Γ, an equivalent conclusion is derivable from $\mathcal{T}(\Gamma)$.

They also show that the translation is *sound*, in the sense that, for each conclusion that may be derived from $\mathcal{T}(\Gamma)$, an equivalent conclusion is derivable from Γ itself. However, this latter statement must be interpreted with a *caveat*: some formulae that can be derived from $\mathcal{T}(\Gamma)$ have in fact no equivalent conclusions derivable from Γ, in the sense that their would-be equivalents are not even expressible in the MINE formalism. For instance, in the case of a MINE Γ such that $C(\Gamma)$ comprises the edge $p \rightarrow q$, the set of formulae derivable from $\mathcal{T}(\Gamma)$ will indeed contain the formula $ISA(p, q)$, but will also contain formulae that do not correspond to conclusions of Γ: among these, formulae containing connectives (such as e.g. $ISA(p, q) \wedge ISA(p, q)$), formulae containing instances of predicates different from ISA and $ISNA$ (such as e.g. $\exists t\ \Pi_P(p, t, q)$), tautologies of the first order predicate calculus, etc. This is an obvious consequence of the fact that we are dealing with a translation of a formalism into an expressively richer one, and does not invalidate the substance of our claim; because of this we have substantially ignored the issue elsewhere in this paper.