Incremental Knowledge Acquisition for Non-Monotonic Reasoning

Fabrizio Sebastiani and 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

Abstract. The use of conventional non-monotonic reasoning tools in real-sized knowledge-based applications is hindered by the fact that the knowledge acquisition (KA) phase cannot be accomplished in the incremental way that is instead typical of knowledge base management systems based on monotonic logics. Some researchers have proposed (non-monotonic) languages for the representation of *Multiple Inheritance Networks with Exceptions* (MINEs) that do not suffer from incrementality problems. However, such languages are formally inadequate, as their semantic status is somewhat questionable. In this paper we discuss an approach to non-monotonic reasoning which does allow the phase of KA to be accomplished incrementally, and 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 vice-versa.

1 Introduction

Incrementality of knowledge acquisition (KA) is an asset of knowledge base (KB) management systems that hardly needs to be argued for. Large KBs 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 KA be *incremental*.

Most "traditional" KB management systems allow for incremental KA because the knowledge representation (KR) languages they rely on are generally

monotonic. Unfortunately, the formalization of many real-sized application domains requires the KR language to include non-monotonic features. Non-monotonic reasoning has been formally addressed in various ways, leading to the development of a variety of formalisms, most of which belong to the offspring of Doyle and McDermott's Nonmonotonic Logic [DM80], Reiter's Default Logic [Rei80] and McCarthy's Circumscription [McC80]. Unfortunately, these formalisms suffer from a problem (that we have dubbed the *Exceptions Explicitation Problem* (EEP)) that makes KA non-incremental: in fact, a (possibly massive) revision of the KB must be operated upon entry of a new knowledge chunk, making the use of such formalisms in substantive KR applications *de facto* impossible.

In the full paper [SS] we discuss, by means of concrete examples, the EEP and how it manifests itself, for example, in the context of Nonmonotonic Logic (NML) (to this respect, other formalisms such as Default Logic and Circumscription behave in a completely analogous way); we also discuss how the addition of an NML formula to a KB calls for a revision of the pre-existing KB that may in general *require repeated calls to the NML theorem prover*, an endeavour that we deem absurd, given the intractability and undecidability of NML. The net effect is that, unless the construction of the KB is realized in a completely static (non incremental) way, the problem of KB construction in NML is practically unsolvable.

While the general non-monotonic formalisms mentioned above are affected by the EEP, this is not true of the languages for the representation of *Multiple Inheritance Networks with Exceptions* (MINEs), a popular, albeit less general, class of non-monotonic KR languages oriented to the representation of taxonomic knowledge. In Section 2 we describe how MINEs do solve the EEP by employing what we call an *implicit handling of exceptions*. Unfortunately, such languages lack a denotational semantics that account for their inferential behaviour in a clear and unambiguous way; therefore, their semantic status is somewhat questionable.

In order to overcome these problems, we are proposing an approach to non-monotonic reasoning that combines the advantages (in terms of incrementality of KA) 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, that we describe in Section 3, 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, besides making KA incremental, is also 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 vice-versa. 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 4 concludes.

2 Multiple Inheritance Networks with Exceptions

A MINE is an acyclic directed graph whose nodes represent classes or individuals of the domain of discourse, and whose edges represent relationships of "conceptual containment" between nodes. 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 inferential mechanisms of these languages are such that an edge $Bird \rightarrow Flies$ has the effect of transmitting "by inheritance" the properties of $Flies$ to $Bird$; this inheritance is *multiple* because a node may inherit properties from different, unrelated nodes. Last, we speak of networks *with exceptions* because the intended meaning of an edge $p \rightarrow q$ is that "p's are typically q's"; this allows for the existence of p's that are not q's, i.e. of exceptions. Dual arguments apply to edges of type $p \not\rightarrow q$.

In MINEs exceptions are *implicitly* handled by the strict partial order induced by the relation $\rightarrow \cup \not\rightarrow$: 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 seen above, we will see how the conclusion that Opus does not fly is given priority, on the grounds that the premise of $Penguin \not\rightarrow Flies$ is more specific than that of $Bird \rightarrow Flies$.

The languages for the representation of MINEs that have been proposed in the literature are by now a fairly vast and multifaceted class (see [SL89] or [THT87] for a review). Our approach substantially relies on the "skeptical" MINE formalism due to Horty *et al.* [HTT90]. For reasons of space, we will describe it only in an informal way; see [SS] for the formal details.

Informally, a MINE Γ is a directed acyclic graph with edges of type $p \rightarrow q$ or $p \not\rightarrow q$; the symbol Γ_E will refer to the set of edges of Γ, while Γ_N will refer to the set of nodes of Γ. Edges concur in the formation of three types of paths: *chains* (i.e. generic concatenations of edges), *positive paths* (i.e. concatenations of "$\rightarrow$" edges) and *negative paths* (i.e. concatenations of "$\rightarrow$" edges followed by a "$\not\rightarrow$" edge). For any chain σ from x to y in Γ, $deg_\Gamma(\sigma)$ will denote the length of the longest chain from x to y in Γ. The *conclusion* that can be drawn from a positive path $x_1 \rightarrow \ldots \rightarrow x_{n-1} \rightarrow x_n$ (resp. from a negative path $x_1 \rightarrow \ldots \rightarrow x_{n-1} \not\rightarrow x_n$) is the edge $x_1 \rightarrow x_n$ (resp. $x_1 \not\rightarrow x_n$).

For example, let us consider a MINE such as $KB1 = \langle \{opus\}, \{Penguin, Bird, Flies, Swims\}, \{opus \rightarrow Penguin, Penguin \rightarrow Bird, Bird \rightarrow Flies, Penguin \not\rightarrow Flies, Flies \not\rightarrow Swims\} \rangle$. From the positive path $opus \rightarrow Penguin \rightarrow Bird \rightarrow Flies$ 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 $opus \rightarrow Penguin \not\rightarrow Flies$ we may conclude $opus \not\rightarrow Flies$ ("As there is no information to the contrary, we assume that Opus does not fly"). This shows that MINEs may 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, one defines which are, given a MINE Γ, the paths

from which "reliable" conclusions may be derived, and that must consequently be given priority over others. For example, in a MINE such as $KB1$ this will allow us to give the path $opus \rightarrow Penguin \not\rightarrow Flies$ priority over the path $opus \rightarrow Penguin \rightarrow Bird \rightarrow Flies$, so inhibiting the undesired conclusion according to which Opus flies. Paths from which we may derive "reliable" conclusions are called *derivable* paths.

Definition 1. 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 $\sigma \in \Gamma_E$ and $x \not\rightarrow y \notin \Gamma_E$;
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 $u \in \Gamma_N$ such that:
   (a) $\sigma = x \rightarrow \delta \rightarrow u \rightarrow y$
   (b) $u \rightarrow y \in \Gamma_E$;
   (c) $\Gamma \vdash x \rightarrow \delta \rightarrow u$;
   (d) $x \not\rightarrow y \notin \Gamma_E$;
   (e) for all nodes v and for all paths τ such that $\Gamma \vdash x \rightarrow \tau \rightarrow v$ and $v \not\rightarrow y \in \Gamma_E$, there exists a node z such that $z \rightarrow y \in \Gamma_E$ 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.

The definition of a *derivable negative path* is dual[1].

By $C(\Gamma)$ we will denote the set of conclusions that may be drawn from the derivable paths of Γ, i.e. the set that represents the knowledge "implicitly" present in the KB.

3 Mapping MINEs into FOL

The implicit handling of exceptions implemented in MINE representation languages allows KA to be incremental: 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 pre-existing 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.

[1] Following the terminology of [THT87] we might say that clauses (b) and (c) of Definition 1 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 3 subscribes to these options, although it might be modified without difficulty in case one wanted to opt for top-down chaining and/or on-path preclusion.

In this paper we are proposing an approach to non-monotonic reasoning that allows for incremental KA, while at the same time relying on a formally solid and widely acknowledged framework such as FOL. 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 KB will be obtained by adding to LT the result of the application of $\mathcal{M}$ to a particular MINE, obtaining the FOL KB $\mathcal{T}(\Gamma) = \mathcal{M}(\Gamma) \cup LT$. The net effect is that the LT component of $\mathcal{T}(\Gamma)$ remains fixed throughout the phase of KA (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 vice-versa. 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 the LT theory, 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 $\mathcal{M}(\Gamma)$[2].

3.1 The $\mathcal{M}$ function

The purpose of the $\mathcal{M}$ function is that of translating a MINE Γ into a set of formulae $\mathcal{M}(\Gamma)$ constituting a first order representation of the graph formed by the nodes and edges of Γ. In order to obtain this, first of all let us represent every node p in Γ_N by an individual constant p of the first order language. The edges of Γ_E 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_E. *Unique names assumptions* are also present in order to reflect their "implicit" presence in the MINE formalism (see [SS] for details). The $\mathcal{M}$ function is then given by the following definition.

Definition 2. Let Γ be a MINE; $\mathcal{M}$ is the function that maps Γ into the FOL axiom

$$(\forall x \forall y\ ISDA(x, y) \Leftrightarrow \bigvee_{p \rightarrow q \in \Gamma_E} (x = p \wedge y = q)) \wedge$$
$$(\forall x \forall y\ ISDNA(x, y) \Leftrightarrow \bigvee_{p \not\rightarrow q \in \Gamma_E} (x = p \wedge y = q)) \wedge$$
$$(\bigwedge_{\{p,q\} \in \Gamma_N} (p \neq q))$$

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 Section 3 we have hinted at the fact that the LT component

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

of $\mathcal{T}(\Gamma)$ remains fixed throughout the phase of KA, while the only variable component is $\mathcal{M}(\Gamma)$. We now see that the way in which $\mathcal{M}(\Gamma)$ 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 KB $\mathcal{T}(\Gamma)$, one needs only to add the formula $(x = r \wedge y = s)$ as a further operand of the disjunction $\bigvee_{p \rightarrow q \in \Gamma_E}(x = p \wedge y = q)$ (resp. $\bigvee_{p \not\rightarrow q \in \Gamma_E}(x = p \wedge y = q)$), and the formula $(r \neq s)$ to the conjunction $\bigwedge_{\{p,q\} \in \Gamma_N}(p \neq q)$. As required, KA becomes a matter of simple piecemeal additions to the pre-existing KB.

3.2 The Logical Theory of Multiple Inheritance with Exceptions

Now that we have defined the $\mathcal{M}$ function, let us build a FOL theory LT such that a formula of type $ISA(x, y)$ (resp. $ISNA(x, y)$) is valid in LT if and only if $x \rightarrow y$ (resp. $x \not\rightarrow y$) belongs to $C(\Gamma)$. The LT theory will consist of the definitions of the predicates ISA and $ISNA$ in terms of the predicates $ISDA$ and $ISDNA$ we have seen in Section 3.1. In order to do so, we will use two predicates $\Pi_P(x, t, y)$ and $\Pi_N(x, t, y)$ representing the derivability in Γ of a (positive or negative, resp.) path from x to y passing from node t. As a consequence, the predicate Π_P must be such that $\Gamma \vdash x \rightarrow \gamma_1 \rightarrow t \rightarrow \gamma_2 \rightarrow y$, for some paths γ_1 and γ_2, iff every model of $\mathcal{M}(\Gamma) \cup LT$ is also a model of $\Pi_P(x, t, y)$. A dual condition must obtain for Π_N.

At this point we may characterize the conclusion that can be drawn from a positive path by adding to LT the simple formula $\forall x \forall y\ (ISA(x, y) \Leftrightarrow \exists t\ \Pi_P(x, t, y))$. The LT theory is then concisely described as follows.

Definition 3. The *Logical Theory of Multiple Inheritance with Exceptions* (LT) is the FOL theory $LT \equiv \{A_1, A_2, A_3, A_4\}$, where[3]:

- $A_1 \equiv \forall x \forall t \forall y\ \Pi_P(x, t, y) \Leftrightarrow$

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

- $A_2 \equiv \forall x \forall y\ (ISA(x, y) \Leftrightarrow \exists t\ \Pi_P(x, t, y))$;

A_3 and A_4 are the dual "negative" versions of A_1 and A_2, respectively.

3.3 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

[3] Axiom A_1 has been subdivided in several numbered subformulae and indented so as to highlight its structural affinity with Definition 1.

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

Theorem 4. *Let Γ be a MINE. Then the following propositions hold*

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

The proof of Theorem 4 is omitted for reasons of space, and is reported only in the full paper [SS].

Clauses (1) and (2) of the theorem 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 derivable 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 whose set of conclusions comprises the edge $p \rightarrow q$, $\mathcal{T}(\Gamma)$ 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 FOL, 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 ignored the issue elsewhere in this paper.

A related comment may be made for clauses (3) and (4) of Theorem 4, in that formulae of the form $\neg ISA(x, y)$ and $\neg ISNA(x, y)$ do not find a direct analogue in the MINE formalism: the result is that the non-derivability of a conclusion (e.g. it is not derivable that birds are typically yellow) finds its translation in the derivability of the negated conclusion (e.g. it is not the case that birds are typically yellow). This can be seen as a *completion* of the theory with respect to the ISA and $ISNA$ predicates.

Finally, clauses (5) and (6) show that ISA and $ISNA$ interact correctly.

In order to better understand the properties of $\mathcal{T}$, we end this section by describing its behaviour in the case of two "classic" examples.

Example 1. Let Γ be a MINE such that $\Gamma_E = \{o \rightarrow P, P \rightarrow B, B \rightarrow F\}$, a MINE that encodes our previous "birds" example. The edge $o \rightarrow F$ belongs to $C(\Gamma)$; in keeping with this, the formula $ISA(o, F)$ is a logical consequence of $\mathcal{T}(\Gamma)$. If we add the edge $P \not\rightarrow F$ to Γ_E, the edge $o \rightarrow F$ no more belongs to $C(\Gamma)$, which

instead includes $o \not\rightarrow F$; accordingly, the formula $ISA(o, F)$ no more belongs to $\mathcal{T}(\Gamma)$, which instead now includes $ISNA(o, F)$. This shows that the $\mathcal{T}$ function is non-monotonic. ∎

Example 2. Let Γ be a MINE such that $\Gamma_E = \{n \rightarrow Q, n \rightarrow R, Q \rightarrow P, R \not\rightarrow P\}$, a MINE that encodes the famous "Nixon diamond". Neither $n \rightarrow P$ nor $n \not\rightarrow P$ belong to $C(\Gamma)$, corresponding to the fact that, according to the "skeptical" approach, whether Nixon is a pacifist or not cannot reasonably be concluded from the information contained in an inherently ambiguous network such as Γ. Accordingly, the formula $\neg ISA(n, P) \wedge \neg ISNA(n, P)$ is a logical consequence of $\mathcal{T}(\Gamma)$. ∎

4 Conclusion

We have described an approach to non-monotonic reasoning that combines the advantages (in terms of incrementality of KA) that are offered by MINE representation languages, with those (in terms of semantic clarity) that are offered by a formally solid apparatus such as First Order Logic. We have achieved this by specifying a first order representation LT of the MINE formalism, and a mapping function $\mathcal{M}$ that maps a MINE Γ into a set of FOL formulae. In our framework, a KB is obtained by adding to LT the result of the application of $\mathcal{M}$ to a particular MINE, thus obtaining the FOL KB $\mathcal{T}(\Gamma) = \mathcal{M}(\Gamma) \cup LT$. The net effect is that the LT component of $\mathcal{T}(\Gamma)$ remains fixed throughout the phase of KA, while the only component that varies is the $\mathcal{M}(\Gamma)$ component. We have shown how this variation is achieved by performing simple piecemeal additions of new chunks of knowledge to the pre-existing KB, thereby accomplishing incrementality and compositionality. We have also shown that $\mathcal{T}$ is a sound and complete mapping, as each conclusion derivable from a MINE Γ is also derivable from $\mathcal{T}(\Gamma)$, and vice-versa.

An interesting side-effect of this result is that MINE-like reasoning can be performed by means of the ordinary and widely available first order theorem provers. A simple interface may be devised that allows the user to convey his information to the KB in terms of the simpler MINE representation language; it is then the interface that takes the charge of implementing the $\mathcal{T}$ translation function, performing the required additive modifications to the axiom of Definition 2. We are currently exploring the possibility of an efficient implementation of this framework by using the "theory resolution" technique for first order theorem proving devised by Stickel [Sti85], which would allow us to "wire" the LT theory into a resolution-based theorem prover, thereby freeing the KB management system from the need to handle the axioms of LT directly.

References

[DM80] Jon Doyle and Drew McDermott. Nonmonotonic logic I. *Artificial Intelligence*, 13:41–72, 1980. [a] Also reprinted in [Gin87], pp. 111–126.

[Gin87] Matthew L. Ginsberg, editor. *Readings in nonmonotonic reasoning*. Morgan Kaufmann, Los Altos, CA, 1987.

[HTT90] John F. Horty, Richmond H. Thomason, and David S. Touretzky. A skeptical theory of inheritance in nonmonotonic semantic networks. *Artificial Intelligence*, 42:311–348, 1990.

[Lif90] Vladimir Lifschitz, editor. *Programs with common sense - Papers by John McCarthy*. Ablex, Norwood, NJ, 1990.

[LNS91] Maurizio Lenzerini, Daniele Nardi, and Maria Simi, editors. *Inheritance hierarchies in knowledge representation and programming languages*. Wiley, Chichester, UK, 1991.

[McC80] John McCarthy. Circumscription - a form of nonmonotonic reasoning. *Artificial Intelligence*, 13:27–39, 1980. [a] Also reprinted in [Gin87], pp. 145–151. [b] Also reprinted in [Lif90], pp. 142–155.

[Rei80] Raymond Reiter. A logic for default reasoning. *Artificial Intelligence*, 13:81–132, 1980. [a] Also reprinted in [Gin87], pp. 68–93.

[SL89] Bart Selman and Hector J. Levesque. The tractability of path-based inheritance. In *Proceedings of IJCAI-89, 11th International Joint Conference on Artificial Intelligence*, pages 1140–1145, Detroit, MI, 1989. [a] Also reprinted in [LNS91], pp. 83–96.

[SS] Fabrizio Sebastiani and Umberto Straccia. Incremental acquisition of knowledge for non-monotonic reasoning (extended report). Technical report, Istituto di Elaborazione dell'Informazione - Consiglio Nazionale delle Ricerche, Pisa, Italy. Forthcoming.

[Sti85] Mark E. Stickel. Automated deduction by theory resolution. *Journal of Automated Reasoning*, 1:333–355, 1985.

[THT87] David S. Touretzky, John F. Horty, and Richmond H. Thomason. A clash of intuitions: the current state of nonmonotonic multiple inheritance systems. In *Proceedings of IJCAI-87, 10th International Joint Conference on Artificial Intelligence*, pages 476–482, Milano, Italy, 1987.