On the Impact of Stratification on the Complexity of Nonmonotonic Reasoning

Ilkka Niemelä — Jussi Rintanen

Helsinki University of Technology
Department of Computer Science
Otakaari 1, FIN–02150 ESPOO, Finland

ABSTRACT. This paper investigates the problem of finding subclasses of nonmonotonic reasoning which can be implemented efficiently. The ability to "define" propositions using default assumptions about the same propositions is identified as a major source of computational complexity in nonmonotonic reasoning. If such constructs are not allowed, i.e. stratified knowledge bases are considered, a significant computational advantage is obtained. This is demonstrated by developing an iterative algorithm for propositional stratified autoepistemic theories the complexity of which is dominated by required classical reasoning. Thus efficient subclasses of stratified nonmonotonic reasoning can be obtained by further restricting the form of sentences in a knowledge base. As an example quadratic and linear time algorithms for specific subclasses of stratified autoepistemic theories are derived. The results are shown to imply efficient reasoning methods for stratified cases of default logic, logic programs, truth maintenance systems, and nonmonotonic modal logics.

KEY WORDS: automated theorem proving, tractability, autoepistemic logic, default logic, nonmonotonic modal logics, logic programs, truth maintenance systems.

---

1. Introduction

Nonmonotonic reasoning is an important aspect of many knowledge representation systems. Databases where the *closed world assumption* [REI 78] is used, logic programming where the *negation as failure* rule [CLA 78] is applied to implement a form of negation, inheritance hierarchies where properties are *inherited by default* [MIN 81], reasoning about action where the *frame*

This is an expanded version of a paper that appeared in the Proceedings of the Third International Conference on Principles of Knowledge Representation and Reasoning (Cambridge, MA, USA, 1992), pp. 627–638.

problem [MCC 69] and the *qualification problem* [MCC 77] have to be solved as well as *diagnosis* [REI 87] are all examples of areas where nonmonotonic reasoning is used. Nonmonotonic reasoning is applied because it is hoped that knowledge representation and reasoning problems can be solved more effectively than using classical (monotonic) reasoning. Recent results suggest that nonmonotonic reasoning is in fact computationally more complex than corresponding classical reasoning [GOT 92, STI 92]. Furthermore, even very restricted subclasses of nonmonotonic reasoning where required classical reasoning is easy turn out to be intractable. Examples of this are simple cases of default logic [KAU 91, STI 90], autoepistemic logic [MAR 91a, MAR 91b], truth maintenance systems [ELK 90], and logic programs [MAR 91a]. Thus reducing the computational complexity of required classical reasoning does not yield the expected efficiency in nonmonotonic reasoning. A typical aspect of the subclasses of nonmonotonic reasoning with disappointing computational properties is that propositions can be "defined" in terms of default assumptions about the same propositions. This seems to result in a situation where finding the correct order of applying defaults (conflict resolution) is computationally very complex.

In this paper we investigate the problem of finding tractable subclasses of nonmonotonic reasoning. An interesting approach is to focus on *stratified* knowledge bases where constructs "defining" propositions using default assumptions about the same propositions are not allowed. The notion of stratification has its origins in the logic programming community [CHA 85, APT 88, VG88] and it has also been studied in the context of other forms of nonmonotonic reasoning such as default logic [BID 87, BID 91a, FRO 88, FRO 92] and autoepistemic logic [GEL 87, MAR 91a]. A knowledge base is stratified if it can be partitioned into a sequence of levels (strata) so that on every level default assumptions are made only about propositions which have already been "defined" on preceding levels. As an example of a stratified knowledge base consider the following logic program representing pieces of knowledge saying that republicans not known to be pacifists are hawks, Rick is a republican and Tom is a pacifist.

$$\text{hawk}(X) \leftarrow \text{republican}(X), \textit{not}\,\text{pacifist}(X). \qquad [1]$$

$$\text{republican(rick)}. \qquad [2]$$

$$\text{pacifist(tom)}. \qquad [3]$$

Consider the partition of the program into two levels where the first level contains the facts 2 and 3 and the second level consists of the rule 1. This partitioning satisfies the condition for stratifiedness as the default assumption (negation as failure) in the rule 1 is made about the predicate pacifist defined already on the first level. Hence the logic program is stratified.

This kind of restriction on the use of default assumptions limits expressivity in the sense that it rules out multiple alternative sets of conclusions: a stratified knowledge base has exactly one possible set of correct conclusions. For example, in the case of logic programs with negation as failure stratification implies a unique canonical model. There are several alternative approaches to characterizing the canonical model of a stratified program [APT 88, VG88, LIF 88, PRZ 88, BID 87, BID 91a, GEL 87, GEL 88b, VG91]. It seems, however, that there are natural situations where competing sets of correct conclusions/canonical models emerge. A typical example is inheritance hierarchies with multiple inheritance. Consider a variant of the "Nixon diamond" [REI 81]. We extend the logic program 1–3 by two rules saying that quakers not known to be hawks are pacifists and that Rick is a quaker.

$$\text{pacifist}(X) \leftarrow \text{quaker}(X), \textit{not}\ \text{hawk}(X). \qquad [4]$$

$$\text{quaker}(\text{rick}). \qquad [5]$$

The resulting program is no longer stratified. Since in the rule defining the predicate pacifist (4) a default assumption is made about the predicate hawk, the rule defining hawk (1) has to appear on some level lower than that defining pacifist (4). However, the rule 1 imposes the opposite constraint on the appearance of rules on the levels. Hence there is no partitioning satisfying the stratifiedness condition. The resulting logic program has two canonical (stable) models. In one Rick is a pacifist but not a hawk and in the other he is a hawk and not a pacifist. The two models seem to capture the essence of the situation: as preference is given to neither of the default assumptions, Rick's status as a pacifist or a hawk remains undetermined although it is known that he is either a pacifist or a hawk. Disjunctive defaults [GEL 91] provide another example where competing sets of correct conclusions seem to be needed and which therefore appears to be outside the realm of stratified knowledge bases. In the treatment of disjunctive defaults a difference is made between "p or q is known" and "p is known or q is known". This difference is captured by using competing sets of extensions (correct conclusions) [GEL 91]. It should be noted that stratification is not a necessary condition for the existence of a single set of correct conclusions. For example, there are non-stratified logic programs with a unique canonical model [GEL 88b, VG91].

We show that stratification offers an interesting trade-off between the expressivity of a knowledge representation language and the computational complexity of reasoning: stratification reduces expressivity but it provides notable computational benefits. In stratified knowledge bases the conflict resolution task can be solved efficiently and the overall complexity of reasoning is dominated by the complexity of the classical reasoning task.

Stratified propositional autoepistemic theories [MAR 91a] are chosen as the basis of the research because autoepistemic logic offers a unified approach to several other types of nonmonotonic reasoning (or at least substantial fragments of these). It has been shown that autoepistemic logic is closely related to default logic [KON 88, MAR 89], circumscription [KON 89], McDermott and Doyle style nonmonotonic modal logics [SHV 90, MAR 93], stable model semantics of logic programs [GEL 88b], nonmonotonic truth maintenance systems [ELK 90], inheritance reasoning [GEL 90], and abduction [KAK 90]. Thus efficient decision methods developed for autoepistemic logic can be applied to other forms of nonmonotonic reasoning.

We develop an iterative algorithm for reasoning in stratified autoepistemic theories. This algorithm provides a general and quite simple iterative decision method for stratified nonmonotonic reasoning including default logic, logic programs, truth maintenance systems, and nonmonotonic modal logics. The algorithm is straightforward to implement and it needs only a decision procedure for the underlying (propositional) reasoning as a subroutine. Furthermore, the coupling of the algorithm and the subroutine is loose. No additional requirements are imposed on the theorem prover used as a subroutine, e.g. with respect to the order in which subgoals are solved. So any decision procedure for the underlying classical reasoning is applicable. We present a detailed analysis of the computational resources needed by the algorithm. This analysis immediately leads to tractable subclasses of nonmonotonic reasoning. As an example we present a quadratic time subclass. By exploiting carefully the logical properties of stratified knowledge bases we are able to devise a linear time decision procedure for a restricted subclass of stratified nonmonotonic reasoning which is still applicable in cases of practical relevance such as stratified logic programs and truth maintenance systems. We are not aware of any decision procedure covering such a large class of stratified knowledge bases that has been given a detailed description and whose computational complexity has been analyzed.

Previous work on computational properties of stratified knowledge bases has mainly addressed stratified logic programs. Kolaitis [KOL 91] has studied the expressive power of stratified logic programs. Kautz and Selman [KAU 91] provide a cubic time algorithm for computing the canonical model of a stratified propositional logic program. Bidoit and Froidevaux [BID 91b] study the class of effectively stratified programs which is a generalization of stratified programs. They develop a method for computing the canonical model of an effectively stratified propositional program which seems to lead to a quadratic algorithm. We improve these results and present a linear time algorithm for computing the canonical model of a stratified program. Lassez et al. [LAS 87] study stratification and knowledge base management and provide various complexity results and algorithms. For example, they present an algorithm for computing a stratification of a stratified knowledge base, i.e. a partitioning of the knowledge base fulfilling the condition for stratifiedness. Marek and Truszczyński [MAR 91a] sketch a stratification algorithm for autoepistemic formulae. Using their ideas we develop a detailed algorithm for deciding whether an autoepistemic set of premises is stratified and for computing a stratification. We show that under certain restrictions on the form of the premises the algorithm runs in linear time.

The rest of the paper is organized as follows. In Section 2 autoepistemic logic is briefly introduced. In Section 3 the notion of stratifiedness is presented and an algorithm for computing a stratification, i.e. a sequence of layers satisfying the stratifiedness condition, is described. In Section 4 an iterative decision method for stratified autoepistemic reasoning is developed. In Section 5 the computational complexity of the decision method is analyzed. Based on the complexity analysis quadratic and linear time algorithms are devised for subclasses of stratified autoepistemic reasoning. Section 6 discusses an implementation of the decision method. Section 7 shows how the decision method can be applied to other forms of nonmonotonic reasoning besides autoepistemic logic and Section 8 contains the concluding remarks.

2. Autoepistemic logic

To obtain the language $\mathcal{L}_{ae}$ of propositional autoepistemic logic we extend the language $\mathcal{L}$ of the propositional calculus by a monadic operator L which is read "is believed". Autoepistemic logic models the beliefs of a fully introspective ideally rational agent. The agent reasons according to a consequence relation $\models$ which is a simple extension of the propositional consequence relation where the $L\phi$ formulae are treated like atomic formulae in the propositional calculus. Given a set of premises a set of correct autoepistemic conclusions is defined as a set of beliefs of the agent with the premises as the initial assumptions of the agent. A set of beliefs is called a *stable expansion* of the premises and it is defined by the following fixed point equation.

Definition 2.1 (Moore [MOO 85]) *Δ is a stable expansion of Σ iff*

$$\Delta = \{\phi \mid \Sigma \cup L\Delta \cup \neg L\overline{\Delta} \models \phi\} \qquad [6]$$

where $L\Delta = \{L\phi \mid \phi \in \Delta\}$, $\neg\Delta = \{\neg\phi \mid \phi \in \Delta\}$, and $\overline{\Delta} = \mathcal{L}_{ae} - \Delta$. Thus $\neg L\overline{\Delta} = \{\neg L\phi \mid \phi \in \mathcal{L}_{ae} - \Delta\}$.

Example 2.1 *Consider a premise $\Sigma = \{\neg Lp \rightarrow q\}$. For Σ there is a unique solution to the fixed point equation 6, i.e. there is a unique stable expansion of Σ. It contains, e.g., $\neg Lp, q, Lq, LLq, L\neg Lp, \neg L\neg Lq, \ldots$* ■

Stable expansions are infinite sets of formulae. A finitary characterization is needed for handling stable expansions computationally. Niemelä [NIE 90] has presented a compact characterization of stable expansions using the notion of a *full set*. The basic idea is to use the $L\phi$ subformulae of the premises to characterize the stable expansions. If the set of premises is finite, the corresponding full sets are finite. In this case infinite stable expansions can be represented finitely.

We use the following notations. For a formula ϕ, $Sf^L(\phi)$ denotes the set of subformulae of the form $L\chi$ of ϕ and $Sf^{qL}(\phi)$ is the set of $L\chi$ quasi-subformulae of ϕ. Quasi-subformulae are subformulae in the usual sense except that $L\chi$ formulae do not have any further quasi-subformulae. For a set of formulae Σ, $Sf^L(\Sigma) = \bigcup_{\phi\in\Sigma} Sf^L(\phi)$ and similarly for $Sf^{qL}(\Sigma)$. For example, for $\Sigma = \{Lp, q \wedge L\neg L(Lp \wedge \neg q)\}$, $Sf^L(\Sigma) = \{Lp, L\neg L(Lp \wedge \neg q), L(Lp \wedge \neg q)\}$ and $Sf^{qL}(\Sigma) = \{Lp, L\neg L(Lp \wedge \neg q)\}$.

Definition 2.2 *A set of formulae Λ is Σ-full if it satisfies the following conditions.*

1. $\Lambda \subseteq Sf^L(\Sigma) \cup \neg Sf^L(\Sigma)$.
2. $L\phi \in \Lambda$ *iff* $\Sigma \cup \Lambda \models \phi$ *for all* $L\phi \in Sf^L(\Sigma)$.
3. $\neg L\phi \in \Lambda$ *iff* $\Sigma \cup \Lambda \not\models \phi$ *for all* $L\phi \in Sf^L(\Sigma)$.

Example 2.2 *Let $\Sigma = \{Lp \rightarrow p, \neg Lp \rightarrow q\}$, where p and q are atomic. By the conditions on full sets for each $L\phi$ subformula of the premises either $L\phi$ or its negation belongs to a full set. Hence, there are two candidates for Σ-full sets: $\Lambda_1 = \{Lp\}$ and $\Lambda_2 = \{\neg Lp\}$. Both are Σ-full. Λ_1 is full as $\Sigma \cup \Lambda_1 \models p$ and Λ_2 is full as $\Sigma \cup \Lambda_2 \not\models p$.* ■

For a set of premises Σ, the Σ-full sets are in a one-to-one correspondence with the stable expansions of Σ [NIE 90]. The unique stable expansion induced by a full set can be characterized with the aid of the consequence relation $\models_L$ which is defined recursively using the underlying consequence relation $\models$. The new consequence relation determines the membership in a stable expansion of Σ when the corresponding full set Λ is known (Definition 4.1 and Theorems 3.15 and 4.2 [NIE 90]):

Definition 2.3 *For $\Sigma \subseteq \mathcal{L}_{ae}$ and $\phi \in \mathcal{L}_{ae}$,*

$$\Sigma \models_L \phi \text{ iff } \Sigma \cup SB_\Sigma(\phi) \models \phi$$

where $SB_\Sigma(\phi) = \{L\chi \in Sf^{qL}(\phi) \mid \Sigma \models_L \chi\} \cup \{\neg L\chi \in \neg Sf^{qL}(\phi) \mid \Sigma \not\models_L \chi\}$.

Theorem 2.4 *Let Λ be a Σ-full set. Then $SE_\Sigma(\Lambda) = \{\phi \mid \Sigma \cup \Lambda \models_L \phi\}$ is the unique stable expansion Δ of Σ such that $\Lambda \subseteq L\Delta \cup \neg L\overline{\Delta}$.*

Example 2.3 *The set of premises Σ in Example 2.2 has two full sets $\{Lp\}$ and $\{\neg Lp\}$. So Σ has exactly two stable expansions $SE_\Sigma(\{Lp\})$ and $SE_\Sigma(\{\neg Lp\})$. The formula $L\neg Lq$ belongs to the former but not to the latter because $\Sigma \cup \{Lp\} \models_L L\neg Lq$ but $\Sigma \cup \{\neg Lp\} \not\models_L L\neg Lq$. For example, $\Sigma \cup \{Lp\} \models_L L\neg Lq$ can be verified as follows. As $SB_{\Sigma\cup\{Lp\}}(q) = \emptyset$ and $\Sigma \cup \{Lp\} \not\models q$, $\Sigma \cup \{Lp\} \not\models_L q$. Thus $SB_{\Sigma\cup\{Lp\}}(\neg Lq) = \{\neg Lq\}$. So $\Sigma \cup \{Lp\} \cup SB_{\Sigma\cup\{Lp\}}(\neg Lq) \models \neg Lq$ which implies $\Sigma \cup \{Lp\} \models_L \neg Lq$. Hence $SB_{\Sigma\cup\{Lp\}}(L\neg Lq) = \{L\neg Lq\}$ and thus $\Sigma \cup \{Lp\} \models_L L\neg Lq$.* ■

3. Stratification

In this section, we first present the definition of stratification for autoepistemic logic as defined by [MAR 91a]. Then we give an algorithm that computes a stratification for a stratified set of formulae or detects that a set is not stratified.

Stratification in autoepistemic logic means that sets of formulae can be partitioned into a number of strata, and beliefs on each stratum refer only to propositional variables defined on strata below. The definition of stratification was introduced to autoepistemic logic by [GEL 87]. In this paper we use a more general definition which is due to Marek and Truszczyński.

Definition 3.1 ([MAR 91a]) *A set of formulae Σ is stratified if*

1. *The formulae $\phi \in \Sigma$ are of the form $a(\phi) \wedge o(\phi) \rightarrow c(\phi)$, where subformulae $o(\phi)$ and $c(\phi)$ do not contain the L operator and $o(\phi)$ may be missing, and $a(\phi)$ is a formula of the form $L\phi_1 \wedge \cdots \wedge L\phi_r \wedge \neg L\psi_1 \wedge \cdots \wedge \neg L\psi_s$ where $r, s \geq 0$.*
2. *The set $\{c(\phi)|\phi \in \Sigma\}$ is satisfiable.*
3. *There exists a set of indices $I = \{1, \ldots, n\}$ or $I = \{1, \ldots\}$ and a partition of $\Sigma = \bigcup_{i \in I} \Sigma_i$ such that for all $j \in I$ the propositional variables occurring in $\{c(\phi)|\phi \in \Sigma_j\}$ do not occur in Σ_j in the scope of an L operator or in $\bigcup_{i=1}^{j-1} \Sigma_i$.*

Example 3.1 *An example of a formula that cannot occur in a stratified set is $L(Lr \vee p) \wedge q \rightarrow p$. This is because of the occurrence of p both in the antecedent inside a L operator and in the consequent. A logically equivalent formula that can occur in a stratified set is $L(Lr \vee p) \wedge \neg p \rightarrow \neg q$.*

The reasoning methods developed in this paper use the stratifications, i.e., the partitions of the formulae in the stratified sets. In the general case the computation of a stratification or testing whether a set of autoepistemic formulae is stratified is computationally very expensive because of the satisfiability testing of $\{c(\phi)|\phi \in \Sigma\}$. This problem is NP-complete; all known algorithms for it take exponential time in the size of the formulae. In the polynomial time classes developed in this paper, Condition 2 of stratification can be tested in linear time. For testing Condition 3 efficiently Marek and Truszczyński define the notion of *a-stratifiedness* which coincides with stratifiedness. A finite set Σ is a-stratified if it satisfies conditions 1 and 2 of stratification and there is a partition $P_1, \ldots, P_n$ of the propositional variables in Σ that fulfills the following condition. For each pair of variables $p \in P_i, q \in P_j$ such that p occurs in $a(\phi)$ and q in $c(\phi)$ for some $\phi \in \Sigma$, $i < j$, and for each pair of variables $p \in P_i, q \in P_j$ such that p occurs in ϕ and q in $c(\phi)$ for some $\phi \in \Sigma$, $i \leq j$.

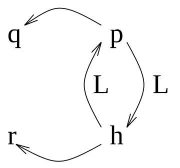
Theorem 3.2 ([MAR 91a]) *A finite set of formulae Σ is stratified if and only if Σ is a-stratified.*

The test for the existence of the partition of propositional variables can easily be reduced to the computation of the strong components of a graph. A strong component is a maximal set of nodes of a graph such that there is a path between any two nodes in the set. The strong components of a graph can be computed in linear time in the size of the graph using Tarjan's well-known algorithm [AHO 74]. The *characteristic graph* of a set of formulae is a graph representing the constraints imposed on the partition of the propositional variables by the definition of a-stratifiedness. In the characteristic graph there is an edge from the variable p to another variable q if for some $\phi \in \Sigma$ p occurs in $c(\phi)$ and q anywhere in ϕ. The edge is an L-edge if the occurrence of q is in $a(\phi)$.

Theorem 3.3 ([MAR 91a]) *A finite set of formulae Σ is a-stratified if and only if Σ satisfies Conditions 1 and 2, and no strong component of its characteristic graph contains an L-edge.*

Example 3.2 *The set*

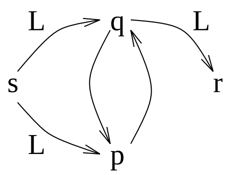
$$r \wedge \neg Lp \rightarrow h$$
$$q \wedge \neg Lh \rightarrow p$$



is not a-stratified and consequently not stratified. The characteristic graph has three strong components. The variables r and q occupy singleton strong components, and since there are edges both from h to p and from p to h, p and h are in the same strong component. The set is not a-stratified – and consequently not stratified – as the edges between p and h are L-edges. ■

Example 3.3 *The set*

$$L\neg Lr \wedge p \rightarrow q$$
$$q \rightarrow p$$
$$L(p \leftrightarrow q) \rightarrow s$$



is a-stratified. Variables p and q belong to the same strong component and there are no L-edges between them, s and r both occupy a singleton strong component. ■

Linear time complexity results in this paper, e.g. the linearity of our stratification algorithm, rest on the following assumption.

Assumption 3.4 *Each propositional variable is assigned a unique number so that data structures with constant access time (arrays) can be used for storing various data related to them.*

The propositional variables occurring in Σ can be assigned unique numbers in $\mathcal{O}(n \log v)$ time, where n is the size of Σ and v is the number of distinct propositional variables occurring in Σ.

For the tractable classes of autoepistemic logic investigated in this paper, we present a linear time algorithm that computes a stratification if the set is stratified, and for sets that are not, detects this fact. The algorithm is based on Theorem 3.3. Marek and Truszczyński [MAR 91a] sketch a similar algorithm for arbitrary sets of autoepistemic formulae that fulfill Condition 1. Another algorithm for computing a stratification which is based on strong components is presented in [LAS 87].

In the general case the size of the characteristic graph of a set Σ is quadratic in the size of Σ, and consequently the traversal of the graph for finding the strong components takes quadratic time. However, if the form of the subformulae $c(\phi), \phi \in \Sigma$ is restricted the computation becomes linear time.

Proposition 3.5 *Let Σ be a set of formulae that fulfills Condition 1 of stratification, and for each $\phi \in \Sigma$, there are occurrences of at most one propositional variable in $c(\phi)$. Under Assumption 3.4 the computation of a stratification for Σ or detecting that Σ is not stratified is $\mathcal{O}(|\Sigma|)$ time.*

Proof: In this restricted case Condition 2 of the definition of a stratified set can be tested in $\mathcal{O}(|\Sigma|)$ time. Because each $c(\phi)$, $\phi \in \Sigma$ contains occurrences of only one propositional variable p, each $c(\phi)$ is equivalent to either p, $\neg p$, $\top$, or $\bot$ as there are exactly four truth functions of one variable. The reduction of all $c(\phi), \phi \in \Sigma$ to one of these formulae is $\mathcal{O}(|\Sigma|)$ time because for each ϕ it can be done in one traversal of the formula tree of $c(\phi)$. The satisfiability of a set of formulae of the form p, $\neg p$, $\bot$, and $\top$ can be tested in linear time in the size of the set. By Theorem 3.3 testing Condition 3 of stratification can be implemented as a computation of the strong components of the characteristic graph together with detection of L-edges inside the strong components. For this purpose we give a variant of Tarjan's [TAR 72] well-known algorithm for the strong components of a graph as presented in [AHO 74]. Tarjan's algorithm runs in $\mathcal{O}(n+e)$ time where n is the number of nodes and e is the number of edges, and it has the useful property that the strong components are produced in an order that qualifies as a stratification, i.e., the first component the algorithm emits consists of the variables in $c(\phi)$ for formulae $\phi \in \Sigma$ that can be taken as the lowest stratum of a stratification $\Sigma_1, \ldots, \Sigma_n$, and so on.

The size of the characteristic graph is linear in the size of Σ because for each $\phi \in \Sigma$ there are occurrences of at most one propositional variable in $c(\phi)$ and consequently there is at most one edge for each variable occurrence in $\{a(\phi) \wedge o(\phi) | \phi \in \Sigma\}$.

For the computation of the strata the following arrays and variables are needed.

formulae$[p]$ the list of formulae ϕ in which the variable p appears in $c(\phi)$. This array can be initialized in linear time by traversing the formulae once.

edges[p] the list of variables q that appear in $a(\phi) \wedge o(\phi)$ for a formula ϕ in which the variable p appears in $c(\phi)$. The initialization can be done in linear time. First initialize the elements to empty lists. Then for each p the list **formulae**[p] is traversed and for each occurrence of a variable q an auxiliary array of flags is tested whether q already is in **edges**[p]. If not, it is added in the head and the auxiliary array is updated.

l-edges[p] the list of variables q that appear in $a(\phi)$ for a formula ϕ in which the variable p appears in $c(\phi)$. This array is initialized in a similar way as the array **edges**.

ccount a counter for the strong components. Initialized to zero.

component[i] the list of formulae in stratum i.

stratum[p] the number of the stratum to which formulae ϕ having p in $c(\phi)$ belong.

The following variables are part of the original strong components algorithm (see [AHO 74] for details):

count a counter for numbering the nodes of the graph in the order of depth-first traversal.

dfnumber[p] the number assigned to the node p during depth-first traversal.

lowlink[p] a number for the node p that is used in recognizing strong components during the traversal.

Before running the algorithm, the above arrays and counters are initialized as indicated above, and all propositional variables occurring in the set Σ are marked *new*. After this, SEARCHC is called repeatedly for *new* variables until all variables are marked *old*. If the error NOT_STRATIFIED is signalled, then Condition 3 cannot be fulfilled and the set is not stratified.

The first half of the procedure is responsible for the traversal of the characteristic graph depth-first and maintenance of the data structures for detection of the strong components. All our modifications are in the *if* statement that forms the second half of the procedure. First, a new element is reserved in the array *components* for the formulae in the newly found strong component. The *repeat* loop assigns each variable in the strong component the number of the component. Finally the *while* loop tests the component for the containment of an L-edge using the numbers assigned to variables, and concatenates to the array *components* the list of formulae belonging to the stratum.

We show that the running time of the algorithm is within bounds proportional to the size of the set of formulae Σ. First note that everything that can be done in $\mathcal{O}(n + e)$ time is within the bounds because the number of nodes, i.e. propositional variables, is bounded by the size of the set of formulae and

```
procedure SEARCHC(v);
begin
  mark v "old";
  dfnumber[v] := count;
  count := count + 1;
  lowlink[v] := dfnumber[v];
  push v on stack;
  for each node w on edges[v] do
    if w is marked "new" then
      SEARCHC(w);
      lowlink[v] := min(lowlink[v],lowlink[w])
    else
      if dfnumber[w] < dfnumber[v] and w is on stack
        then lowlink[v] := min(dfnumber[w],lowlink[v])
      end if
    end if
  end for;
  if lowlink[v] = dfnumber[v] then
    ccount := ccount + 1;
    components[ccount] := empty_list;
    initialize stack2 to empty;
    repeat
      pop x from top of stack;
      push x to stack2;
      stratum[x] := ccount
    until x=v;
    while stack2 not empty do
      pop x from stack2;
      for each y in l-edges[x] do
        if stratum[x] = stratum[y]
          then signal NOT_STRATIFIED
        end if
      end for;
      concatenate formulae[x] to components[ccount]
    end while
  end if
end
```

Figure 1. *The main procedure of an algorithm for computing a stratification*

so is the number of edges which is the number of variable occurrences in the antecedents of formulae.

SEARCHC is called exactly once for each node in the graph and each call (the recursive call to SEARCHC and the second half of the procedure excluded) takes constant time plus time proportional to the number of edges leaving from the node. For the whole run of the algorithm this is $\mathcal{O}(n+e)$. All computation in the second half of the procedure during the *whole* computation of the algorithm is bounded by the size of the set of formulae. The update of *ccount* and initialization of the *components*[*ccount*] and the stack are constant time operations and they are performed for each strong component found. In one run of the algorithm exactly one iteration of the *repeat* loop is executed for each node of the graph, and this is $\mathcal{O}(|\Sigma|)$. The tests for containment of L-edges are also within the $\mathcal{O}(|\Sigma|)$ bound because for each propositional variable p the presence of L-edges inside the stratum of p is tested exactly once, and the number of elements in the lists of the array *l-edges* is linearly bounded by the size of the set of formulae. Constructing the lists of the array *components* is $\mathcal{O}(|\Sigma|)$ since each formula belongs to exactly one stratum and we restrict $c(\phi)$ for each ϕ to contain occurrences of at most one propositional variable. □

Example 3.4 *Our algorithm computes for the set in Example 3.3 the stratification shown below. The first column contains the numbers of the strata, the second contains the variables in the corresponding strong components of the associated characteristic graph, and the third the strata, i.e., the sets of formulae ϕ for which the variables of the respective strong component occur in $c(\phi)$.*

3	s	$L(p \leftrightarrow q) \rightarrow s$
2	p, q	$q \rightarrow p, L\neg Lr \wedge p \rightarrow q$
1	r	$\emptyset$

Because there are no formulae ϕ with r in $c(\phi)$, the lowest stratum is empty and can be ignored. ■

4. A decision procedure for stratified theories

In this section we develop an iterative decision method for stratified sets of premises. In autoepistemic reasoning the correct conclusions are given in terms of stable expansions of a set of premises. Each stratified set of premises has a unique stable expansion and thus the notion of correct conclusions is unambiguously defined: a formula is an autoepistemic conclusion from a stratified set of premises if the formula belongs to the unique stable expansion of the premises.

Theorem 2.4 implies the following approach to automating stratified autoepistemic reasoning. Given a stratified set of premises Σ compute first the full set Λ for the unique expansion of Σ and then decide whether the given formula belongs to the corresponding stable expansion $SE_{\Sigma}(\Lambda)$ by using the $\models_L$ consequence relation. It turns out that the full set can be computed iteratively using a stratification of the premises. First the $L\phi$ subformulae appearing in the lowest stratum in the stratification are considered, then the next stratum and so on.

The next theorem gives the basic iterative algorithm for computing the Σ-full set of an arbitrary stratified set Σ. After the correctness proof of the algorithm we develop a more efficient version where the full set is not explicitly constructed (Theorem 4.5).

Theorem 4.1 *Let $\Sigma = \bigcup_{k=1}^{n} \Sigma_k$ be a stratified set. Then $\Lambda = \Lambda_n$ defined by*

$$\begin{aligned}
\Lambda_0 &= \emptyset \\
\Lambda_{i+1} &= \Lambda_i \cup \{L\chi \mid L\chi \in Sf^L(\Sigma_{i+1}) - Sf^L(\bigcup_{k=1}^{i} \Sigma_k), \bigcup_{k=1}^{i} \Sigma_k \cup \Lambda_i \models_L \chi\} \cup \\
&\quad \{\neg L\chi \mid L\chi \in Sf^L(\Sigma_{i+1}) - Sf^L(\bigcup_{k=1}^{i} \Sigma_k), \bigcup_{k=1}^{i} \Sigma_k \cup \Lambda_i \not\models_L \chi\}, 0 \leq i < n
\end{aligned}$$

is Σ-full and $SE_{\Sigma}(\Lambda)$ is the unique stable expansion of Σ.

For proving Theorem 4.1 the following lemma is essential. It is the most important piece for the correctness of the algorithm in the theorem, since it accounts for the monotonicity property that allows the conclusion of underivability of formulae, and hence an iterative instead of a backtracking algorithm. From now on, we use the notation Σ_a^b for the union $\bigcup_{i=a}^{b} \Sigma_i$.

Lemma 4.2 *Let $\Sigma = \Sigma_1^n$ be stratified and $\Lambda_i, i \in \{1, \ldots, n\}$, as in Theorem 4.1. Then for all $i, 0 \leq i < n$ and for all $L\chi \in Sf^L(\Sigma_{i+1})$, $\Sigma_1^i \cup \Lambda_i \models_L \chi$ iff $\Sigma_1^j \cup \Lambda_j \models_L \chi$ iff $\Sigma_1^j \cup \Lambda_j \models \chi$, where $j > i$ and $j \leq n$.*

Proof: By induction on i.

($i = 0$). The proof that for all $L\chi \in Sf^L(\Sigma_1)$, $\Sigma_1^0 \cup \Lambda_0 \models_L \chi$ iff $\Sigma_1^j \cup \Lambda_j \models_L \chi$, and $\Sigma_1^0 \cup \Lambda_0 \models_L \chi$ iff $\Sigma_1^j \cup \Lambda_j \models \chi$ is by induction on the L-depth s of χ. The L-depth of a formula is the maximum nesting of L operators in it.

($s = 0$). For all $L\chi \in Sf^L(\Sigma_1)$ with L-depth 0,

$$\begin{aligned}
\Sigma_1^0 \cup \Lambda_0 \models_L \chi \quad &\text{iff} \quad \Sigma_1^0 \cup \Lambda_0 \models \chi && [7] \\
&\text{implies} \quad \Sigma_1^j \cup \Lambda_j \models \chi && [8] \\
&\text{iff} \quad \Sigma_1^j \cup \Lambda_j \models_L \chi. && [9]
\end{aligned}$$

Equivalence 7 is because $SB_{\Delta}(\chi) = \emptyset$ for χ with no L operators, and consequently $\models$ coincides with $\models_L$ for χ. Implication 8 is because $\models$ is monotonic and Σ_1^0 and Λ_0 are empty by definition. Equivalence 9 is because $\models$ coincides with $\models_L$ for χ.

To establish the equivalence between $\Sigma_1^0 \cup \Lambda_0 \models \chi$ and $\Sigma_1^j \cup \Lambda_j \models \chi$ we have to prove the converse of implication 8 above. This is proved by showing that $\emptyset \not\models \chi$ implies $\Sigma_1^j \cup \Lambda_j \not\models \chi$. Assume $\emptyset \not\models \chi$, i.e., there is a model $\mathcal{M}$ such that $\mathcal{M} \not\models \chi$. We show that then also $\Sigma_1^j \cup \Lambda_j \not\models \chi$ holds by giving a model $\mathcal{M}'$ such that $\mathcal{M}' \models \Sigma_1^j \cup \Lambda_j$ and $\mathcal{M}' \not\models \chi$. The model $\mathcal{M}'$ can be obtained by modifying $\mathcal{M}$ to satisfy $\Sigma_1^j \cup \Lambda_j$. We show that this modification is possible and that it does not make χ true in $\mathcal{M}'$. The set Σ_1^j can be made true in $\mathcal{M}'$ by making the formulae $\{c(\phi) | \phi \in \Sigma_1^j\}$ true. This set of formulae is satisfiable by the second condition of stratification. The truth-value of χ is not affected because by the third condition of stratification the set of propositional variables in $L\chi \in Sf^L(\Sigma_1)$ is disjoint from the variables in $\{c(\phi) | \phi \in \Sigma_1^j\}$. Making the formulae in Λ_j true does not affect the valuation of χ or $\{c(\phi) | \phi \in \Sigma_1^j\}$ because the valuation of formulae beginning with L is disjoint from propositional variables. The set Λ_j is satisfiable because by construction it contains no formula and its negation.

($s \geq 1$). For all $L\chi \in Sf^L(\Sigma_1)$ with L-depth s,

$$\begin{array}{llll}
\Sigma_1^0 \cup \Lambda_0 \models_L \chi & \text{iff} & \Sigma_1^0 \cup \Lambda_0 \cup SB_{\Sigma_1^0 \cup \Lambda_0}(\chi) \models \chi & [10] \\
& \text{iff} & SB_{\Sigma_1^0 \cup \Lambda_0}(\chi) \models \chi & [11] \\
& \text{iff} & SB_{\Sigma_1^j \cup \Lambda_j}(\chi) \models \chi & [12] \\
& \text{implies} & \Sigma_1^j \cup \Lambda_j \cup SB_{\Sigma_1^j \cup \Lambda_j}(\chi) \models \chi & [13] \\
& \text{iff} & \Sigma_1^j \cup \Lambda_j \models \chi & [14] \\
& \text{iff} & \Sigma_1^j \cup \Lambda_j \models_L \chi. & [15]
\end{array}$$

Equivalence 10 is by the definition of $\models_L$. Equivalence 11 is because by definition $\Sigma_1^0 \cup \Lambda_0$ is empty. Equivalence 12 is by Lemma **A** below. Implication 13 is by the monotonicity of $\models$. Equivalence 14 is because $SB_{\Sigma_1^j \cup \Lambda_j}(\chi) = SB_{\Sigma_1^0 \cup \Lambda_0}(\chi)$ is contained in Λ_j as shown by Lemma **B** below. Equivalence 15 is by Lemma **B** and the definition of $\models_L$.

Lemma A. For all $j, 1 \leq j < n$, $SB_{\Sigma_1^0 \cup \Lambda_0}(\chi) = SB_{\Sigma_1^j \cup \Lambda_j}(\chi)$.
Formulae ϕ such that $L\phi \in Sf^{qL}(\chi)$ are in $Sf^L(\Sigma_1)$ because $L\chi \in Sf^L(\Sigma_1)$, and thus by the induction hypothesis on s we get $\Sigma_1^0 \cup \Lambda_0 \models_L \phi$ iff $\Sigma_1^j \cup \Lambda_j \models_L \phi$, for formulae $L\phi \in Sf^{qL}(\chi)$. By the definition of SB the claim is immediate. □

Lemma B. $SB_{\Sigma_1^0 \cup \Lambda_0}(\chi) \subseteq \Lambda_j$.
Suppose $L\phi \in SB_{\Sigma_1^0 \cup \Lambda_0}(\chi)$, i.e. $L\phi \in Sf^{qL}(\chi)$ and $\Sigma_1^0 \cup \Lambda_0 \models_L \phi$. Because $L\chi \in Sf^L(\Sigma_1)$ also $L\phi \in Sf^L(\Sigma_1)$. Now $L\phi \in \Lambda_1 \subseteq \Lambda_j, j \geq 1$ by the equations of Theorem 4.5. Similarly with $\neg L\phi$. □

To establish the equivalence between $\Sigma_1^0 \cup \Lambda_0 \models_L \chi$ and $\Sigma_1^j \cup \Lambda_j \models_L \chi$ we have to prove the converse of implication 13 above. This is proved by showing that $SB_{\Sigma_1^j \cup \Lambda_j}(\chi) \not\models \chi$ implies $\Sigma_1^j \cup \Lambda_j \cup SB_{\Sigma_1^j \cup \Lambda_j}(\chi) \not\models \chi$. Assume $SB_{\Sigma_1^j \cup \Lambda_j}(\chi) \not\models \chi$, i.e., there is a model $\mathcal{M}$ such that $\mathcal{M} \models SB_{\Sigma_1^j \cup \Lambda_j}(\chi)$ and

$\mathcal{M} \not\models \chi$. We show that then also $\Sigma_1^j \cup \Lambda_j \cup SB_{\Sigma_1^j \cup \Lambda_j}(\chi) \not\models \chi$ holds by giving a model $\mathcal{M}'$ such that $\mathcal{M}' \models \Sigma_1^j \cup \Lambda_j \cup SB_{\Sigma_1^j \cup \Lambda_j}(\chi)$ and $\mathcal{M}' \not\models \chi$. The model $\mathcal{M}'$ can be obtained by modifying $\mathcal{M}$ to satisfy $\Sigma_1^j \cup \Lambda_j$. We show that this modification is possible and that it does not make χ true in $\mathcal{M}'$. The set Σ_1^j can be made true in $\mathcal{M}'$ by making the formulae $\{c(\phi)|\phi \in \Sigma_1^j\}$ true. This set of formulae is satisfiable by the second condition of stratification. The truth-value of χ is not affected because by the third condition of stratification the set of propositional variables in $L\chi \in Sf^L(\Sigma_1)$ is disjoint from the variables in $\{c(\phi)|\phi \in \Sigma_1^j\}$. Making the formulae in Λ_j true does not affect the valuation of $\{c(\phi)|\phi \in \Sigma_1^j\}$ because the valuation of formulae beginning with L is disjoint from propositional variables. The set Λ_j is satisfiable because by construction it contains no formula and its negation. The valuation of Λ_j does not conflict with the valuation of $SB_{\Sigma_1^j \cup \Lambda_j}(\chi) = SB_{\Sigma_1^0 \cup \Lambda_0}(\chi)$ because the latter is contained in Λ_j as shown by Lemma **B** above. Consequently, because the valuation of formulae $L\psi \in Sf^L(\chi)$ does not change, the truth-value of χ is not affected.

($i \geq 1$). Proof is by induction on the L-depth s of χ.

($s = 0$). For all $L\chi \in Sf^L(\Sigma_{i+1})$ with L-depth 0,

$$
\begin{aligned}
\Sigma_1^i \cup \Lambda_i \models_L \chi \quad & \text{iff} \quad \Sigma_1^i \cup \Lambda_i \models \chi && [16]\\
& \text{implies} \quad \Sigma_1^j \cup \Lambda_j \models \chi && [17]\\
& \text{iff} \quad \Sigma_1^j \cup \Lambda_j \models_L \chi. && [18]
\end{aligned}
$$

Equivalence 16 is because $SB_\Delta(\chi) = \emptyset$ for χ with no L operators, and consequently $\models$ coincides with $\models_L$ for χ. Implication 17 is because $\Lambda_i \subseteq \Lambda_j, \Sigma_1^i \subseteq \Sigma_1^j$ for $i \leq j$, and $\models$ is monotonic. Equivalence 18 is again because $\models$ coincides with $\models_L$ for χ.

To establish the equivalence between $\Sigma_1^i \cup \Lambda_i \models_L \chi$ and $\Sigma_1^j \cup \Lambda_j \models_L \chi$ we have to prove the converse of implication 17 above. This is proved by showing that $\Sigma_1^i \cup \Lambda_i \not\models \chi$ implies $\Sigma_1^j \cup \Lambda_j \not\models \chi$. Assume $\Sigma_1^i \cup \Lambda_i \not\models \chi$, i.e., there is a model $\mathcal{M}$ such that $\mathcal{M} \models \Sigma_1^i \cup \Lambda_i$ and $\mathcal{M} \not\models \chi$. We show that then also $\Sigma_1^j \cup \Lambda_j \not\models \chi$ holds by giving a model $\mathcal{M}'$ such that $\mathcal{M}' \models \Sigma_1^j \cup \Lambda_j$ and $\mathcal{M}' \not\models \chi$. The model $\mathcal{M}'$ can be obtained by modifying $\mathcal{M}$ to satisfy $\Sigma_{i+1}^j \cup \Lambda_j$. We show that this modification is possible and that it does not make χ true in $\mathcal{M}'$. The set Σ_{i+1}^j can be made true in $\mathcal{M}'$ by making the formulae $\{c(\phi)|\phi \in \Sigma_{i+1}^j\}$ true. This set of formulae is satisfiable by the second condition of stratification. The truth-value of χ is not affected because by the third condition of stratification the set of propositional variables in $L\chi \in Sf^L(\Sigma_i)$ is disjoint from the variables in $\{c(\phi)|\phi \in \Sigma_{i+1}^j\}$. Making the formulae in $\Lambda_j - \Lambda_i$ true does not affect the valuation of χ or $\{c(\phi)|\phi \in \Sigma_1^j\}$ because the valuation of formulae beginning with L is disjoint from propositional variables. The set $\Lambda_j - \Lambda_i$ is satisfiable because by construction it contains no formula and its negation.

($s \geq 1$). For all $L\chi \in Sf^L(\Sigma_{i+1})$,

$$
\Sigma_1^i \cup \Lambda_i \models_L \chi \qquad \text{iff} \qquad \Sigma_1^i \cup \Lambda_i \cup SB_{\Sigma_1^i \cup \Lambda_i}(\chi) \models \chi \qquad [19]
$$

$$
\begin{aligned}
&\text{iff} && \Sigma_1^i \cup \Lambda_i \cup SB_{\Sigma_1^j \cup \Lambda_j}(\chi) \models \chi && [20]\\
&\text{implies} && \Sigma_1^j \cup \Lambda_j \cup SB_{\Sigma_1^j \cup \Lambda_j}(\chi) \models \chi && [21]\\
&\text{iff} && \Sigma_1^j \cup \Lambda_j \models \chi && [22]\\
&\text{iff} && \Sigma_1^j \cup \Lambda_j \models_L \chi. && [23]
\end{aligned}
$$

Equivalence 19 is by the definition of $\models_L$. Equivalence 20 is by Lemma **C** below. Implication 21 is because $\Lambda_i \subseteq \Lambda_j, \Sigma_1^i \subseteq \Sigma_1^j$ for $i \leq j$, and $\models$ is monotonic. Equivalence 22 is because $SB_{\Sigma_1^i \cup \Lambda_i}(\chi) = SB_{\Sigma_1^j \cup \Lambda_j}(\chi)$ is contained in Λ_j as shown by Lemmata **C** and **D** below. Equivalence 23 is by Lemma **D** and the definition of $\models_L$.

Lemma C. For all $j, i < j < n$, $SB_{\Sigma_1^i \cup \Lambda_i}(\chi) = SB_{\Sigma_1^j \cup \Lambda_j}(\chi)$.
Formulae ϕ such that $L\phi \in Sf^{qL}(\chi)$ are in $Sf^L(\Sigma_{i+1})$ as $L\chi \in Sf^L(\Sigma_{i+1})$, and thus by the induction hypothesis on s we get $\Sigma_1^i \cup \Lambda_i \models_L \phi$ iff $\Sigma_1^j \cup \Lambda_j \models_L \phi$, for formulae $L\phi \in Sf^{qL}(\chi)$. By the definition of SB the claim is immediate. □

Lemma D. For all $j, i < j \leq n$, $SB_{\Sigma_1^i \cup \Lambda_i}(\chi) \subseteq \Lambda_j$.
Assume $L\phi \in SB_{\Sigma_1^i \cup \Lambda_i}(\chi)$.

1.	$\chi \in Sf^L(\Sigma_{i+1})$	assumption
2.	$L\phi \in SB_{\Sigma_1^i \cup \Lambda_i}(\chi)$	assumption
3.	$L\phi \in Sf^{qL}(\chi)$	2 and definition of SB
4.	$\Sigma_1^i \cup \Lambda_i \models_L \phi$	2 and definition of SB
5.	$L\phi \in Sf^L(\Sigma_{i+1})$	1 and 3
6.	$\exists k \leq i, L\phi \in Sf^L(\Sigma_{k+1}) - Sf^L(\Sigma_1^k)$	5
7.	$\Sigma_1^k \cup \Lambda_k \models_L \phi$	4, 6, induction hypothesis on i
8.	$L\phi \in \Lambda_{k+1} \subseteq \Lambda_j$	7, equations of Theorem 4.5

Similarly with $\neg L\phi$. □

To establish the equivalence between $\Sigma_1^i \cup \Lambda_i \models_L \chi$ and $\Sigma_1^j \cup \Lambda_j \models_L \chi$ we have to prove the converse of implication 21 above. This is proved by showing that $\Sigma_1^i \cup \Lambda_i \cup SB_{\Sigma_1^j \cup \Lambda_j}(\chi) \not\models \chi$ implies $\Sigma_1^j \cup \Lambda_j \cup SB_{\Sigma_1^j \cup \Lambda_j}(\chi) \not\models \chi$. Assume $\Sigma_1^i \cup \Lambda_i \cup SB_{\Sigma_1^j \cup \Lambda_j}(\chi) \not\models \chi$, i.e., there is a model $\mathcal{M}$ such that $\mathcal{M} \models \Sigma_1^i \cup \Lambda_i \cup SB_{\Sigma_1^j \cup \Lambda_j}(\chi)$ and $\mathcal{M} \not\models \chi$. We show that then also $\Sigma_1^j \cup \Lambda_j \cup SB_{\Sigma_1^j \cup \Lambda_j}(\chi) \not\models \chi$ holds by giving a model $\mathcal{M}'$ such that $\mathcal{M}' \models \Sigma_1^j \cup \Lambda_j \cup SB_{\Sigma_1^j \cup \Lambda_j}(\chi)$ and $\mathcal{M}' \not\models \chi$. The model $\mathcal{M}'$ can be obtained by modifying $\mathcal{M}$ to satisfy $\Sigma_{i+1}^j \cup \Lambda_j$. We show that this modification is possible and that it does not make χ true in $\mathcal{M}'$. The set Σ_{i+1}^j can be made true in $\mathcal{M}'$ by making the formulae $\{c(\phi) | \phi \in \Sigma_{i+1}^j\}$ true. This set of formulae is satisfiable by the second condition of stratification. The truth-value of χ is not affected because by the third condition of stratification the set of propositional variables in $L\chi \in Sf^L(\Sigma_i)$ is disjoint from the variables in $\{c(\phi) | \phi \in \Sigma_{i+1}^j\}$. Making the formulae in $\Lambda_j - \Lambda_i$ true does not affect the valuation of $\{c(\phi) | \phi \in \Sigma_1^j\}$ because the valuation of formulae beginning with L is disjoint from propositional variables. The set

Λ_j is satisfiable because by construction it contains no formula together with its negation. The valuation of $\Lambda_j - \Lambda_i$ does not conflict with the valuation of $SB_{\Sigma_1^i \cup \Lambda_j}(\chi) = SB_{\Sigma_1^i \cup \Lambda_i}(\chi)$ because the latter set is contained in Λ_j as shown by Lemma **D** above. Consequently, because the valuation of formulae $L\psi \in Sf^L(\chi)$ does not change, the truth-value of χ is not affected. $\square$

Proof of Theorem 4.1: Λ is Σ-full because it satisfies the conditions of Definition 2.2. Condition 1 is immediate, and Condition 3 is true if Condition 2 is, since by construction for all $L\chi \in Sf^L(\Sigma)$ exactly one of $L\chi$ or $\neg L\chi$ is in Λ. Condition 2 holds because $L\chi \in \Lambda$ if and only if there is $i < n$ for which $L\chi \in (Sf^L(\Sigma_{i+1}) - Sf^L(\Sigma_1^i))$ and $\Sigma_1^i \cup \Lambda_i \models_L \chi$. By Lemma 4.2 this is equivalent to $\Sigma_1^n \cup \Lambda \models \chi$. By Theorem 2.4 $SE_\Sigma(\Lambda)$ is a stable expansion of the set of premises Σ, and because Σ is stratified $SE_\Sigma(\Lambda)$ is the unique stable expansion of Σ by Theorem 5.1 of [MAR 91a]. $\square$

The following two lemmata are needed for Theorem 4.5 which shows how the unique stable expansion of a stratified set Σ can be computed more efficiently without explicitly constructing the Σ-full set Λ.

Lemma 4.3 *Let Σ be a set of formulae of the form $L\chi_1 \wedge \cdots \wedge L\chi_n \wedge \neg L\chi_{n+1} \wedge \cdots \wedge \neg L\chi_{n+m} \wedge \psi \rightarrow \psi'$ where $\psi, \psi' \in \mathcal{L}$, and let Λ be a set of formulae of the form $L\phi, \neg L\phi$ that contains exactly one of $L\chi$, $\neg L\chi$ for each $L\chi \in Sf^L(\Sigma)$. Define $Red(\Sigma, \Lambda) = \{\psi \rightarrow \psi' | (\phi \wedge \psi \rightarrow \psi') \in \Sigma$, the conjuncts of ϕ are in $\Lambda\}$. Assume that for all $L\phi, \neg L\phi' \in \Lambda$, $\Sigma \cup \Lambda \models_L \phi$ and $\Sigma \cup \Lambda \not\models_L \phi'$. Then for all $\chi \in \mathcal{L}_{ae}$,*

$$Red(\Sigma, \Lambda) \models_L \chi \textit{ iff } \Sigma \cup \Lambda \models_L \chi.$$

Proof: The equivalence $\phi \wedge \psi \rightarrow \psi' \equiv \phi \rightarrow (\psi \rightarrow \psi')$ justifies proving the lemma using $Red(\Sigma, \Lambda) = \{\psi | (\phi \rightarrow \psi) \in \Sigma$, the conjuncts of ϕ are in $\Lambda\}$. The proof is by induction on the L-depth s of χ.

($s = 0$). For formulae with L-depth 0 there are no L subformulae and therefore $\models_L$ coincides with $\models$. ($\Rightarrow$). The implication $Red(\Sigma, \Lambda) \models_L \chi$ implies $\Sigma \cup \Lambda \models_L \chi$ is shown by contraposition, i.e., $\Sigma \cup \Lambda \not\models_L \chi$ implies $Red(\Sigma, \Lambda) \not\models_L \chi$. Suppose $\Sigma \cup \Lambda \not\models \chi$, i.e., there is a model $\mathcal{M}$ such that $\mathcal{M} \models \Sigma \cup \Lambda$ and $\mathcal{M} \not\models \chi$. We show that also $\mathcal{M} \models Red(\Sigma, \Lambda)$ and hence $Red(\Sigma, \Lambda) \not\models \chi$. Suppose $(\phi \rightarrow \psi) \in \Sigma$. If $\psi \in Red(\Sigma, \Lambda)$, then the conjuncts of ϕ are in Λ and consequently $\mathcal{M} \models \phi$. Because $\mathcal{M} \models \phi \rightarrow \psi$ also $\mathcal{M} \models \psi$. This shows that $\mathcal{M} \models Red(\Sigma, \Lambda)$, and $Red(\Sigma, \Lambda) \not\models \chi$.

($\Leftarrow$). Suppose $Red(\Sigma, \Lambda) \not\models \chi$, i.e. there is a model $\mathcal{M}$ such that $\mathcal{M} \models Red(\Sigma, \Lambda)$ and $\mathcal{M} \not\models \chi$. Let $\mathcal{M}'$ be $\mathcal{M}$ modified to satisfy Λ. Because there are no subformulae in $Red(\Sigma, \Lambda)$ or in χ that begin with L, the formulae in Λ can be made true without affecting truth-values of $Red(\Sigma, \Lambda)$ or χ. Hence $\mathcal{M}' \models \Lambda$ and $\mathcal{M}' \not\models \chi$. For showing that $\mathcal{M}' \models \Sigma$ the analysis for formulae $(\phi \rightarrow \psi) \in \Sigma$ is divided to two cases. First, if all conjuncts of ϕ are in Λ then ψ is in $Red(\Sigma, \Lambda)$ and therefore $\mathcal{M} \models \psi$, and further, $\mathcal{M}' \models \phi \rightarrow \psi$. Second, if at least one conjunct of ϕ is not in Λ then ϕ is false in $\mathcal{M}'$ (because then the complement of the conjunct is in Λ) and again $\mathcal{M}' \models \phi \rightarrow \psi$. Therefore $\Sigma \cup \Lambda \not\models_L \chi$.

($s \geq 1$). ($\Rightarrow$) Suppose $\Sigma \cup \Lambda \not\models_L \chi$, i.e., there is a model $\mathcal{M}$ such that $\mathcal{M} \models \Sigma \cup \Lambda \cup SB_{\Sigma\cup\Lambda}(\chi)$ and $\mathcal{M} \not\models \chi$. We show that also $\mathcal{M} \models Red(\Sigma,\Lambda) \cup SB_{Red(\Sigma,\Lambda)}(\chi)$ and hence $Red(\Sigma,\Lambda) \not\models_L \chi$. By the induction hypothesis $SB_{Red(\Sigma,\Lambda)}(\chi) = SB_{\Sigma\cup\Lambda}(\chi)$, and hence $\mathcal{M} \models SB_{Red(\Sigma,\Lambda)}(\chi)$. Suppose $(\phi \rightarrow \psi) \in \Sigma$. If $\psi \in Red(\Sigma,\Lambda)$, then the conjuncts of ϕ are in Λ and consequently $\mathcal{M} \models \phi$. Because $\mathcal{M} \models \phi \rightarrow \psi$ also $\mathcal{M} \models \psi$. This shows that $\mathcal{M} \models Red(\Sigma,\Lambda)$, and finally $Red(\Sigma,\Lambda) \not\models \chi$.

($\Leftarrow$). Suppose $Red(\Sigma,\Lambda) \not\models_L \chi$, i.e. there is a model $\mathcal{M}$ such that $\mathcal{M} \models Red(\Sigma,\Lambda) \cup SB_{Red(\Sigma,\Lambda)}(\chi)$ and $\mathcal{M} \not\models \chi$. Let $\mathcal{M}'$ be $\mathcal{M}$ modified to satisfy Λ. This modification does not affect the truth-value of $Red(\Sigma,\Lambda)$ because there are no formulae beginning with L in $Red(\Sigma,\Lambda)$. That $SB_{Red(\Sigma,\Lambda)}(\chi)$ is true in $\mathcal{M}'$ is seen from the following chain of equivalences. For ϕ such that $L\phi \in Sf^{qL}(\chi)$ and $L\phi$ or $\neg L\phi$ in Λ,

$$
\begin{aligned}
L\phi \in SB_{Red(\Sigma,\Lambda)}(\chi) \quad &\text{iff} \quad L\phi \in SB_{\Sigma\cup\Lambda}(\chi) && [24]\\
&\text{iff} \quad \Sigma \cup \Lambda \models_L \phi && [25]\\
&\text{iff} \quad L\phi \in \Lambda && [26]
\end{aligned}
$$

and for both $L\phi \in Sf^{qL}(\chi)$ and Λ the formula $\neg L\phi$ is in the set exactly when $L\phi$ is not. Equivalence 24 is by the induction hypothesis. Equivalence 25 is by the definition of *SB*. Equivalence 26 is by the assumption of the lemma.

The truth value of χ is the same in $\mathcal{M}'$ and $\mathcal{M}$ because the truth-values of propositional variables are the same in both of these models, and the subformulae $Sf^{qL}(\chi)$ of χ have the same truth values in both models as already shown above by the argument concerning $SB_{\Sigma\cup\Lambda}(\chi)$. Hence $\mathcal{M}' \models \Lambda \cup SB_{\Sigma\cup\Lambda}(\chi)$ and $\mathcal{M}' \not\models \chi$.

The argument for $\mathcal{M}' \models \Sigma$ is the same as in the base case of the induction. □

Lemma 4.4 *Let ϕ be of the form $L\phi_1 \wedge \cdots \wedge L\phi_n \wedge \neg L\psi_1 \wedge \cdots \wedge \neg L\psi_m$ and $\Sigma \subseteq \mathcal{L}$. Then $\Sigma \models_L \phi$ iff $\Sigma \models_L \phi_i$ for all $i, 1 \leq i \leq n$ and $\Sigma \not\models_L \psi_j$ for all $j, 1 \leq j \leq m$.*

Proof: ($\Rightarrow$) Suppose that for some i, $\Sigma \not\models_L \phi_i$, or for some j, $\Sigma \models_L \psi_j$. Then one of $L\phi_i$ or $\neg L\psi_j$ is not in $SB_\Sigma(\phi)$ and $\Sigma \not\models_L \phi$. ($\Leftarrow$) Suppose that the antecedent is true. Then $SB_\Sigma(\phi) = \{L\phi_1, \ldots, L\phi_n, \neg L\psi_1, \ldots, \neg L\psi_m\}$, and therefore $\Sigma \cup SB_\Sigma(\phi) \models \phi$ and $\Sigma \models_L \phi$. □

Let Λ be the Σ-full set corresponding to the unique stable expansion $SE_\Sigma(\Lambda)$ of a stratified set Σ. The following theorem gives an algorithm for computing directly the set $Red(\Sigma,\Lambda) \subseteq \mathcal{L}$ which characterizes the stable expansion $SE_\Sigma(\Lambda)$.

Theorem 4.5 *Let $\Sigma = \Sigma_1^n$ be a stratified set, and define $Red_L(\Sigma, R) = \{o(\phi) \rightarrow c(\phi) | \phi \in \Sigma, R \models_L a(\phi)\}$. Define $R = R_n$ by*

$$\begin{aligned} R_0 &= \emptyset \\ R_{i+1} &= R_i \cup Red_L(\Sigma_{i+1}, R_i), 0 \leq i < n. \end{aligned}$$

Then $R = Red(\Sigma, \Lambda)$ and $\{\phi | R \models_L \phi\} = SE_\Sigma(\Lambda)$ where Λ is Σ-full.

Proof: Let Λ be the set computed by the algorithm of Theorem 4.1 for a set Σ. Because Λ is Σ-full it fulfills the conditions of Lemma 4.3, and hence the stable expansion of Σ is $SE_\Sigma(\Lambda) = \{\phi | Red(\Sigma, \Lambda) \models_L \phi\}$. We show by induction that for all $i, 0 \leq i \leq n$, $R_i = Red(\Sigma_1^i, \Lambda_i)$.

($i = 0$) Immediate, as $R_0 = \emptyset = Red(\Sigma_1^0, \Lambda_0)$.

($i \geq 1$).

$$\begin{aligned} R_i &= R_{i-1} \cup Red_L(\Sigma_i, R_{i-1}) \\ &= Red(\Sigma_1^{i-1}, \Lambda_{i-1}) \cup Red_L(\Sigma_i, R_{i-1}) \\ &= Red(\Sigma_1^{i-1}, \Lambda_{i-1}) \cup Red(\Sigma_i, \Lambda_i) \\ &= Red(\Sigma_1^i, \Lambda_i) \end{aligned}$$

The first equality is by the definition of R_i and the second by the induction hypothesis. The fourth is because $\Lambda_{i-1} \subseteq \Lambda_i$ and for each $L\phi \in Sf^L(\Sigma_1^{i-1})$ there is either $L\phi$ or $\neg L\phi$ in Λ_{i-1}. The third equality is shown as follows. Let $\phi \in \Sigma_i$. Then by Lemma 4.4 $o(\phi) \rightarrow c(\phi) \in Red_L(\Sigma_i, R_{i-1})$ iff for all conjuncts $L\chi$ in $a(\phi)$, $R_{i-1} \models_L \chi$, and for all conjuncts $\neg L\chi$ in $a(\phi)$, $R_{i-1} \not\models_L \chi$. Consider conjuncts $L\chi$ in $a(\phi)$:

$$\begin{aligned} R_{i-1} \models_L \chi \quad &\text{iff} \quad Red(\Sigma_1^{i-1}, \Lambda_{i-1}) \models_L \chi \\ &\text{iff} \quad \Sigma_1^{i-1} \cup \Lambda_{i-1} \models_L \chi \\ &\text{iff} \quad \exists k \leq i, L\chi \in (Sf^L(\Sigma_k) - Sf^L(\Sigma_1^{k-1})), \Sigma_1^{k-1} \cup \Lambda_{k-1} \models_L \chi \\ &\text{iff} \quad L\chi \in \Lambda_k \subseteq \Lambda_i. \end{aligned}$$

The first equivalence is by the induction hypothesis. The second is by Lemma 4.3. The third is because either $i = k$ and the equivalence is immediate, or $k < i$ and $L\chi \in Sf^L(\Sigma_k)$, and it follows by Lemma 4.2. The fourth equivalence is by the definition of Λ_j. Similarly for conjuncts $\neg L\chi$ in $a(\phi)$. By the definition of *Red* and the equivalences above, $\phi \in Red_L(\Sigma_i, R_{i-1})$ iff $\phi \in Red(\Sigma_i, \Lambda_i)$. □

Example 4.1 *The table below demonstrates the computation of the full set Λ of the set Σ in Example 3.3 by the algorithm of Theorem 4.1, and the computation of $R = Red(\Sigma, \Lambda)$ by the algorithm of Theorem 4.5.*

i	Σ_i	Λ	R
2	$L(p \leftrightarrow q) \rightarrow s$	$L(p \leftrightarrow q)$	s
1	$L\neg Lr \wedge p \rightarrow q$	$\neg Lr, L\neg Lr$	$p \rightarrow q$
	$q \rightarrow p$		$q \rightarrow p$

■

5. Complexity results

For discussing the complexity of decision problems of autoepistemic logic we briefly introduce some basic concepts from [GAR 79]. The polynomial hierarchy is an infinite hierarchy of complexity classes Σ_i^p, Π_i^p, and Δ_i^p, $i \geq 0$ defined using oracle Turing machines in the following way.

$$\begin{array}{rclcrclcrcl} \Sigma_0^p &=& P & \quad & \Pi_0^p &=& P & \quad & \Delta_0^p &=& P \\ \Sigma_{n+1}^p &=& \mathrm{NP}^{\Sigma_n^p} & & \Pi_{n+1}^p &=& \text{co-}\Sigma_{n+1}^p & & \Delta_{n+1}^p &=& P^{\Sigma_n^p} \end{array}$$

$C_1^{C_2}$ denotes the class of problems that belong to the class C_1 if given an oracle for a problem in C_2, and co-C denotes the set of problems whose complements are in the class C. Note that Σ_1^p =NP. A problem is *C-hard* if all members of the class C can be transformed to it in polynomial time. A problem is *C-complete* if it belongs to the class C and is C-hard.

In the most general cases, the decision problems of propositional autoepistemic logic lie on the second level of the polynomial hierarchy. *Brave reasoning*, the membership of a formula in at least one stable expansion of a set of formulae, is a Σ_2^p-complete problem, and *cautious reasoning*, the membership of a formula in all stable expansions of a set of formulae, is a Π_2^p-complete problem [GOT 92]. In these cases there are two sources of complexity, the classical propositional reasoning and the problem of finding stable expansions, both of which bring the complexity up one level in the polynomial hierarchy. Marek and Truszczyński [MAR 91b, MAR 91a] have investigated the complexity of autoepistemic reasoning in restricted cases. They show that for sets of formulae of the form $Lp_1 \wedge \cdots \wedge Lp_n \wedge \neg Lq_1 \wedge \cdots \wedge \neg Lq_m \rightarrow r$, where p_i, q_j and r are propositional variables, the decision problems of brave and cautious reasoning are NP-complete and co-NP-complete, respectively [MAR 91b]. Even for sets of formulae as simple as $\neg Lp \rightarrow q$ the existence of stable expansions remains NP-complete [MAR 91a].

The method developed in the previous section shows that when making the restriction to stratified sets of formulae, there is only one source of complexity that in practise requires exponential time computation, namely the classical reasoning. This is established in Theorem 5.2 which gives bounds for the complexity of stratified propositional autoepistemic logic in the general case. As a basis of analyzing the complexity of stratified autoepistemic logic, we use the algorithm in Theorem 4.5. The algorithm computes a set $R \subseteq \mathcal{L}$ that characterizes the unique stable expansion of a stratified set. By the next lemma the explicit construction of the sets $SB_\Sigma(\chi)$ can be avoided in the $\models_L$ tests of the algorithm. As a result, all consequence tests are for sets of formulae in which no L operators occur.

Lemma 5.1 *Let $\chi(L\phi)$ be a formula in which the formula $L\phi$ possibly occurs and $\chi(\top)$ the same formula where all occurrences of $L\phi$ have been replaced by the constant true $\top$. Let $\Sigma \subseteq \mathcal{L}_{ae}$ be a set of formulae where $L\phi$ does not* *occur. Now $\Sigma \cup \{L\phi\} \models \chi(L\phi)$ iff $\Sigma \models \chi(\top)$, and $\Sigma \cup \{\neg L\phi\} \models \chi(L\phi)$ iff $\Sigma \models \chi(\bot)$, where $\bot = \neg\top$.*

Proof: ($\Leftarrow$) Suppose $\Sigma \cup \{L\phi\} \not\models \chi(L\phi)$, i.e., there is a model M such that $\mathcal{M} \models \Sigma \cup \{L\phi\}$ and $\mathcal{M} \not\models \chi(L\phi)$. Clearly $\mathcal{M} \not\models \chi(\top)$. ($\Rightarrow$) Suppose $\Sigma \not\models \chi(\top)$, i.e., $\mathcal{M} \models \Sigma$ and $\mathcal{M} \not\models \chi(\top)$. Since the truth value of $L\phi$ is independent of other formulae of the form $L\psi$ and of propositional variables, $\top$ can be replaced by $L\phi$ in χ and therefore $\Sigma \cup \{L\phi\} \not\models \chi(L\phi)$. Similarly for $\neg L\phi$ and $\bot$. $\square$

Theorem 5.2 *In propositional stratified autoepistemic logic, brave and cautious reasoning are co-NP-hard and in Δ_2^p.*

Proof: Because of the uniqueness of stable expansions in stratified autoepistemic logic, the problems of brave and cautious reasoning coincide. A stratification for a stratified set Σ can be found in polynomial time using the algorithm given in the previous section. The algorithm given in Theorem 4.5 computes for Σ a set R such that $\{\phi \in \mathcal{L}_{ae} | R \models_L \phi\}$ is the unique stable expansion of Σ. In its computation, the algorithm tests the consequence $\Sigma' \models_L a(\phi)$ for each member ϕ of Σ and some set $\Sigma' \subseteq \mathcal{L}$. Each such consequence test reduces to tests for logical consequence in propositional logic, the number of which is proportional to the length of $a(\phi)$. This is by the use of Lemma 5.1. Hence the total number of logical consequence tests – which can be performed by one call to an oracle for the satisfiability of propositional logic – is linear in the size of Σ. The membership of a formula ϕ in the unique stable expansion characterized by the set R can be tested by $R \models_L \phi$. This test reduces to a linear number of calls to the NP oracle for propositional satisfiability. Hence the decision problems are in Δ_2^p. The co-NP-complete problem of propositional validity can be reduced in polynomial time to the problem of membership in the unique stable expansion of the stratified set $\emptyset$. Hence the decision problems are co-NP-hard. $\square$

In this section we analyze the complexity of our algorithm in cases where the time complexity of the classical reasoning component is polynomial. The proof of Theorem 5.2 indicates that restricting the classical reasoning to a subclass that can be decided in polynomial time brings the time complexity of the decision problems of stratified propositional autoepistemic logic to polynomial. The basis of the analysis is the algorithm given in Theorem 4.5. In this algorithm all logical consequence tests are for objective formulae and can be implemented as calls to a theorem prover for classical propositional logic. This is very convenient from the point of view of implementing a theorem prover for stratified autoepistemic logic because existing theorem provers for classical logic can be employed as subroutines without modification.

The complexity analysis gives a formula for the amount of resources (time or space) parameterized with the complexity $R(x)$ of the theorem prover employed. The satisfiability of propositional Horn clauses can be tested in linear time, and by assigning $R(x) = ax + c$ for linear time, a quadratic time upper bound is obtained for the membership testing in the stable expansions of a class of stratified propositional autoepistemic theories based on Horn clauses. For a more restricted class based on program clauses we observe a specific property that allows membership tests in linear time.

We write $|\phi|$ for the length of a formula ϕ, $|\Sigma|$ for the sum of lengths of the formulae in a set Σ, and $||\Sigma||$ for the cardinality of a set Σ.

Lemma 5.3 *$F(\chi, \Sigma)$ in Equation 27 gives the amount of resources needed for testing the $\models_L$-consequence of an arbitrary formula χ from a set of formulae $\Sigma \subseteq \mathcal{L}$. $R(x)$ is the amount of resources needed for performing a consequence test of size x.*

$$F(\chi, \Sigma) = R(|\chi| + |\Sigma| - \sum_{L\phi \in Sf^{qL}(\chi)} |\phi|) + \sum_{L\phi \in Sf^{qL}(\chi)} F(\phi, \Sigma) \qquad [27]$$

and the equation with recursion removed is

$$\begin{aligned} F(\chi, \Sigma) &= R(|\chi| + |\Sigma| - \sum_{L\phi \in Sf^{qL}(\chi)} |\phi|) \\ &\quad + \sum_{L\phi \in Sf^{L}(\chi)} R(|\phi| + |\Sigma| - \sum_{L\psi \in Sf^{qL}(\phi)} |\psi|). \end{aligned} \qquad [28]$$

Proof: The second summand of the right hand side of Equation 27 corresponds to the computation of the members of $SB_\Sigma(\chi)$ using $\models_L$. The first summand corresponds to the consequence test $\Sigma \models_L \chi$, or equivalently the consequence test $\Sigma \cup SB_\Sigma(\chi) \models \chi$. By Lemma 5.1 the consequence test can be made by replacing $L\phi$ in χ by $\top$ for all $L\phi \in SB_\Sigma(\chi)$, and by $\perp$ for all $\neg L\phi \in SB_\Sigma(\chi)$, thus obtaining $\chi' \in \mathcal{L}$ and then testing the consequence $\Sigma \models \chi'$. This is why $\sum_{L\phi \in Sf^{qL}(\chi)} |\phi|$ is subtracted. □

Next we give an upper bound for the amount of resources needed for consequence tests in the computation of the algorithm in Theorem 4.5. Let $\Sigma = \Sigma_1^n$ be a stratified set and let there be an enumeration of the formulae $\phi_i \in \Sigma, 1 \leq i \leq r$ such that if the stratum of ϕ_i is lower than that of ϕ_j then $i < j$. Let $n_i = |a(\phi_i)|$ and $m_i = |o(\phi_i) \rightarrow c(\phi_i)|$. Ignoring the exact boundaries between the strata is an acceptable approximation since we are primarily interested in analyzing the upper bounds of complexity. The size of a stratified set is the sum of the sizes of its formulae

$$\sum_{i=1}^{r} (n_i + m_i + 1). \qquad [29]$$

An upper bound for the resources needed for the consequence tests is

$$\sum_{i=1}^{r} F(a(\phi_i), \bigcup_{j=1}^{i-1} \{o(\phi_j) \rightarrow c(\phi_j)\}). \qquad [30]$$

Tractable classes of stratified sets of autoepistemic formulae can be found by restricting the syntactic form of the formulae in such a way that the classical theorem proving task becomes tractable. As an example we present a tractable class SHC_{ae} based on Horn clauses, i.e. disjunctions of literals of which at most one is positive. The class is designed so that all $\models_L$ consequence tests in the algorithm of Theorem 4.5 use only such $\models$ consequence tests that reduce to satisfiability testing of sets of Horn clauses. Satisfiability of propositional Horn clauses can be tested in linear time by using the algorithm of Dowling and Gallier [DOW 84]. Hence the algorithm in Theorem 4.5 runs in this case in polynomial time.

Definition 5.4 *A formula χ is in the class HF_{ae} if it is a disjunction of conjunctions of formulae of the form p, $\neg p$, $L\phi$, and $\neg L\phi$ with at most one $\neg p$ in each disjunct, where each p is a propositional variable and each ϕ is in HF_{ae}.*

Definition 5.5 *SHC_{ae} is the class of finite stratified sets of formulae ϕ of the form $a(\phi) \wedge o(\phi) \rightarrow c(\phi)$ where $a(\phi)$ is a conjunction of zero or more formulae of the form $L\chi$ or $\neg L\chi$ where χ is in HF_{ae}, $o(\phi)$ is a conjunction of zero or more propositional variables, and $c(\phi)$ is a propositional variable or a negated propositional variable.*

The following are examples of formulae in sets in SHC_{ae}.

$$a \wedge b \rightarrow c$$
$$\neg L(LLp \vee q \vee \neg r \vee (t \wedge \neg u) \vee (x \wedge y)) \rightarrow \neg a$$

Theorem 5.6 *For a set Σ in SHC_{ae} the set $Red(\Sigma, \Lambda)$, where Λ is the unique Σ-full set, can be computed in $\mathcal{O}(n^2)$ time, where $n = |\Sigma|$.*

Proof: By Proposition 3.5 a stratification can be determined in $\mathcal{O}(|\Sigma|)$ time. The computation of $R = Red(\Sigma, \Lambda)$ by the algorithm in Theorem 4.5 is dominated by the consequence tests: in the computation of $Red_L(\Sigma_{i+1}, R_i)$ the removal of whole formulae in Σ_{i+1}, the removal of the autoepistemic parts $a(\phi), \phi \in \Sigma_{i+1}$, and the replacement of formulae $L\phi$ by $\top$ or $\perp$ during a $\models_L$-consequence test, are all constant time operations, and the number of these operations is smaller than the size of Σ_{i+1}.

All consequence tests reduce to satisfiability tests for formulae in conjunctive normal form with at most one positive literal in each conjunct, and hence the linear time algorithm of [DOW 84] can be used. For logical consequence tests in the computation of the algorithm in Theorem 4.5, instantiate $R(x) = ax + c$ to the second equation of Lemma 5.3:

$$\begin{aligned} F(\chi, \Sigma) &= a(|\chi| + |\Sigma| - \sum_{L\phi \in Sf^{qL}(\chi)} |\phi|) + c \\ &\quad + \sum_{L\phi \in Sf^{L}(\chi)} (a(|\phi| + |\Sigma| - \sum_{L\psi \in Sf^{qL}(\phi)} |\psi|) + c). \end{aligned}$$

By reordering the terms we obtain

$$\begin{aligned} F(\chi,\Sigma) &= a(|\chi| + |\Sigma| + \sum_{L\phi\in Sf^L(\chi)} |\Sigma|) + (c + \sum_{L\phi\in Sf^L(\chi)} c) \\ &\quad + a(-\sum_{L\phi\in Sfq^L(\chi)} |\phi| + \sum_{L\phi\in Sf^L(\chi)} |\phi| - \sum_{L\phi\in Sf^L(\chi)} \sum_{L\psi\in Sfq^L(\phi)} |\psi|). \end{aligned}$$

The value of the third summand is zero since the length of each $L\phi \in Sf^L(\chi)$ is added and subtracted exactly once. From this we get the upper bound

$$\begin{aligned} &a(|\chi| + |\Sigma| + \|Sf^L(\chi)\| \cdot |\Sigma|) + c(1 + \|Sf^L(\chi)\|) \\ &\leq a(|\chi| + |\Sigma| \cdot (1 + \|Sf^L(\chi)\|)) + c \cdot |\chi|) \\ &\leq a(|\chi| + |\Sigma| \cdot |\chi|) + c \cdot |\chi| \\ &< (a+c)(|\chi| + |\Sigma| \cdot |\chi|) \end{aligned}$$

for $F(\chi,\Sigma)$. Using this and Equation 30 (by the definition of $\mathcal{O}$ the constant factor $a+c$ appearing in each term can be left out) we get

$$\begin{aligned} \sum_{i=1}^{r}(n_i + n_i \sum_{j=1}^{i-1} m_i) &\leq (\sum_{i=1}^{r} n_i) + (\sum_{i=1}^{r} n_i)(\sum_{i=1}^{r} m_i) \\ &\leq (\sum_{i=1}^{r} n_i)^2 + 2(\sum_{i=1}^{r} n_i)(\sum_{i=1}^{r} m_i) + (\sum_{i=1}^{r} m_i)^2 \\ &< (\sum_{i=1}^{r}(n_i + m_i + 1))^2. \end{aligned}$$

This establishes the $\mathcal{O}(n^2)$ upper bound for the logical consequence tests. □

Theorem 5.7 *Given $Red(\Sigma,\Lambda)$ where $\Sigma \in SHC_{ae}$ and Λ is a Σ-full set, the membership in the unique stable expansion $\{\phi | Red(\Sigma,\Lambda) \models_L \phi\}$ of Σ for formulae ϕ in HF_{ae} can be decided in $\mathcal{O}(n^2)$ time, where $n = |\Sigma| + |\neg\phi|$.*

Proof: By the second equation of Lemma 5.3 the amount of resources needed is bounded by $|\Sigma| \cdot |\neg\phi| + |\neg\phi|$ as shown in the proof of Theorem 5.6, and this is $\mathcal{O}(n^2)$. □

Theorem 5.8 *Let Σ be in SHC_{ae}. The membership problem of the unique stable expansion of Σ for formulae ϕ in HF_{ae} is solvable in $\mathcal{O}(n^2)$ time, where $n = |\Sigma| + |\neg\phi|$.*

The line taken in establishing the above result suggests further tractable classes based on other subsets of propositional logic for which polynomial time satisfiability tests are available, like those presented in [SCH 78, GAL 88]. Next we investigate an even more restricted class for which the logical consequence testing does not have to be done separately for each formula inside L.

Definition 5.9 *A formula χ is in CF_{ae} if it is a conjunction of one or more formulae of the form p, $L\phi$, and $\neg L\phi$ where each p is a propositional variable and each ϕ is in CF_{ae}.*

Definition 5.10 *SPC_{ae} is the class of finite stratified sets of formulae ϕ of the form $a(\phi) \wedge o(\phi) \rightarrow c(\phi)$ where $a(\phi)$ is a conjunction of zero or more formulae of the form $L\chi$ or $\neg L\chi$ and each χ is in CF_{ae}, $o(\phi)$ is a conjunction of zero or more propositional variables, and $c(\phi)$ is a propositional variable.*

The following formulae illustrate the form of formulae in sets in SPC_{ae}:

$$\neg L(a \wedge b \wedge c \wedge \neg Ld) \rightarrow e$$
$$L(p \wedge Lq \wedge \neg Lr) \wedge \neg L(\neg Ls) \wedge t \rightarrow u$$

The class SPC_{ae} is a proper subclass of SHC_{ae}. Formulae in SHC_{ae} are more expressive than those in SPC_{ae} in the sense that in SHC_{ae} it is possible to use a negative literal in the head of an implication and disjunctions and negative literals are allowed inside L operators. For example, formulae

$$\neg Lp \rightarrow \neg a$$
$$L(q \vee r) \rightarrow b$$
$$L\neg p \rightarrow c$$
$$L(\neg t \vee s) \wedge \neg L(LL\neg p \vee p \vee \neg r \vee (t \wedge \neg u) \vee (x \wedge y)) \rightarrow \neg d$$

could be used in a stratified set belonging to SHC_{ae} but not in a set in SPC_{ae}.

In [DOW 84] two linear time algorithms are given for testing the satisfiability of a set of Horn clauses. We modify one of these to be used in linear time tests for the membership in the unique stable expansions of sets in SPC_{ae}. By Theorem 4.5 explicit construction of the sets Λ_i can be avoided and instead of separately testing the logical consequence of each formula inside L in $a(\phi)$ the whole set of consequences for one stratum of a stratification can be computed by one run of the algorithm.

The function DG is the basis of the efficient algorithm for SPC_{ae} and can be implemented as a variant of Algorithm 2 of [DOW 84]. Dowling and Gallier's algorithm works with arbitrary Horn clauses, but ours is restricted to *program clauses*, i.e., disjunctions of literals exactly one of which is positive. Sometimes program clauses are written as $a_1 \wedge \cdots \wedge a_n \rightarrow b$ instead of $\neg a_1 \vee \cdots \vee \neg a_n \vee b$.

In our algorithm sets of propositional variables are represented by their characteristic functions or equivalently arrays. For array indexing Assumption 3.4 is essential.

Proposition 5.11 *Let Σ be a set of program clauses and v a set of propositional variables. Then*

$$DG(\Sigma, v) = \{p | p \text{ is a propositional variable, } \Sigma \cup v \models p\}$$

can be computed in $\mathcal{O}(|\Sigma|)$ time.

```
function DG(Σ, v : array of {0,1}) : array of {0,1};
begin
   initialize poslitlist, clauselist, numargs;
   initialize queue to the list of clauses n for which numargs[n] = 0;
   for each c in queue do v(poslitlist[c]) := 1;
   while queue is not empty do
     clause1 := pop(queue);
     for each clause2 in clauselist[clause1] do
       numargs[clause2] := numargs[clause2] - 1;
       if numargs[clause2] = 0 then
         n := poslitlist[clause2];
         if v(n) = 0 then
           v(n) := 1;
           queue := push(clause2,queue)
         end if
       end if
     end for
   end while;
   return v
end
```

Figure 2. *A version of Algorithm 2 of Dowling and Gallier*

Proof: The computation of DG for a set of program clauses Σ and a set of propositional variables v uses the following arrays:

poslitlist Each element n is initialized to the propositional variable appearing in the positive literal of the clause $\phi_n \in \Sigma$. The initialization can be done by one traversal of the set of formulae, which is linear time.

clauselist The element n is the list of clauses of Σ in which the variable in the positive literal of ϕ_n appears in a negative literal. Initialization can be done in linear time.

numargs The element n is initialized to the number of variables p in the negative literals of ϕ_n for which $v(p) = 0$.

The function is computed by the procedure in Figure 2. Under Assumption 3.4 the computation is linear time in the size of Σ. The argument is the same as for the linearity of Algorithm 2 of [DOW 84]. The initializations of the procedure for a set of clauses Σ is $\mathcal{O}(|\Sigma|)$. Each clause n of Σ is pushed to the queue at most once, i.e. in the initialization part of the procedure when it is found that $v(p) = 1$ for all body variables p that appear negatively in n, or inside the *while* loop when the value of $v(poslitlist[n])$ is changed from 0 to 1. Each round of the *while* loop corresponds to the deletion of negative occurrences of some propositional variable $poslitlist[c]$, and for each propositional variable there is at most one round of the loop. Hence the amount of computation inside the while loop is proportional to the number of negative literals in Σ. $\Box$

Define $Red_P(\Sigma, v) = \{\phi \rightarrow \phi' | (L\chi_i \wedge \cdots \wedge L\chi_n \wedge \neg L\chi_{n+1} \wedge \cdots \wedge \neg L\chi_{n+m} \wedge \phi \rightarrow \phi') \in \Sigma, v \models_L \chi_1, \ldots, v \models_L \chi_n, v \not\models_L \chi_{n+1}, \ldots, v \not\models_L \chi_{n+m}\}$. The set $Red_P(\Sigma, v)$ can be computed in linear time in $|\Sigma|$ for members of SPC$_{ae}$. The consequence tests $v \models_L \chi$, χ in CF$_{ae}$ can be done in linear time in $|\chi|$ by using the following algorithm.

$$v \models_L \chi \text{ iff for each conjunct } \phi \text{ of } \chi \begin{cases} \text{if } \phi \text{ is atomic then } \phi \in v \\ \text{if } \phi = L\psi \text{ then } v \models_L \psi \\ \text{if } \phi = \neg L\psi \text{ then } v \not\models_L \psi \end{cases}$$

The following two lemmata are needed for establishing Lemma 5.14 which describes how to compute in $\mathcal{O}(|\Sigma|)$ time the set of propositional variables in the unique stable expansion of a set Σ in SPC$_{ae}$.

Lemma 5.12 *Let $\Sigma \subseteq \mathcal{L}$, $v = \{p | p$ is a propositional variable, $\Sigma \models p\}$, and χ be in CF$_{ae}$. Then $v \models_L \chi$ iff $\Sigma \models_L \chi$.*

Proof: By induction on the L-depth s of χ. ($s = 0$). Suppose $\Sigma \not\models_L \chi$, i.e., there is a model $\mathcal{M}$ for which $\mathcal{M} \models \Sigma$ and $\mathcal{M} \not\models \chi$. Because v is the set of propositional variables true in every model of Σ, also $\mathcal{M} \models v$, and $v \not\models_L \chi$. Suppose $\Sigma \models_L \chi$. Because χ is a conjunction of propositional variables each of which is a logical consequence of Σ, obviously $v \models_L \chi$. ($s \geq 1$). By the induction hypothesis $SB_v(\chi) = SB_\Sigma(\chi)$, and replacing members of $SB_v(\chi)$ (and $SB_\Sigma(\chi)$) according to Lemma 5.1 by $\top$ or $\bot$, χ can be reduced to χ' of L-depth 0 for which $v \models \chi'$ iff $\Sigma \models \chi'$, which is shown as in the case $s = 0$. $\Box$

Lemma 5.13 *Let Σ be a set of program clauses and Γ the set of propositional variables p such that $\Sigma \models p$. Let Δ be a set of program clauses such that the propositional variables in the positive literals of Δ do not occur in the negative literals of Σ. Then for all propositional variables p, $\Sigma \cup \Delta \models p$ iff $\Gamma \cup \Delta \models p$.*

Proof: Suppose $\Sigma \cup \Delta \not\models p$, i.e., for some model $\mathcal{M}$, $\mathcal{M} \models \Sigma \cup \Delta$ and $\mathcal{M} \not\models p$. Clearly $\mathcal{M} \models \Gamma \cup \Delta$ and therefore $\Gamma \cup \Delta \not\models p$. Suppose $\Gamma \cup \Delta \not\models p$, i.e., there is a model $\mathcal{M}$, $\mathcal{M} \models \Gamma \cup \Delta$, $\mathcal{M} \not\models p$. $\mathcal{M}'$ is $\mathcal{M}$ modified to satisfy $\Sigma \cup \Delta$ in the following way. Formulae $\phi = \neg a_1 \vee \cdots \vee \neg a_n \vee b, \phi \in \Sigma$ can be false if $\mathcal{M} \models a_1 \wedge \cdots \wedge a_n$ and $\mathcal{M} \not\models b$. Now $\mathcal{M}'$ can be modified to make ϕ true by making one of $a_i, 1 \leq i \leq n$ false. This can be done without falsifying formulae in $\Gamma \cup \Delta$ because **i)** not all $a_i, 1 \leq i \leq n$ can be logical consequences of Σ and hence members of Γ. If they were, then $b \in \Gamma$. And **ii)** as propositional variables in the positive literals of Δ do not occur in the negative literals of Σ, this modification falsifies no formula in Δ. Therefore $\mathcal{M}' \models \Sigma \cup \Delta$ and $\Sigma \cup \Delta \not\models p$. $\Box$

Lemma 5.14 *For $\Sigma = \Sigma_1^n$ in SPC_{ae} and a formula χ in CF_{ae}, $v_n \models_L \chi$ iff χ belongs to the unique stable expansion of Σ, where v_n is defined by*

$$\begin{aligned} v_0 &= \emptyset \\ v_{i+1} &= DG(Red_P(\Sigma_{i+1}, v_i), v_i), 0 \leq i < n. \end{aligned}$$

Furthermore, v_n, which is the set of propositional variables in the unique stable expansion of Σ, can be computed in $\mathcal{O}(|\Sigma|)$ time.

Proof: We prove $v_i \models_L \chi$ iff $Red(\Sigma_1^i, \Lambda_i) \models_L \chi$, where Λ_i as in Theorem 4.1, by induction on i. ($i = 0$). Immediate as $v_0 = Red(\Sigma_1^0, \Lambda_0) = \emptyset$. ($i > 0$). For all propositional variables p,

$$\begin{aligned} v_i \models p \quad &\text{iff} \quad v_{i-1} \cup Red_P(\Sigma_i, v_{i-1}) \models p \\ &\text{iff} \quad Red(\Sigma_1^{i-1}, \Lambda_{i-1}) \cup Red_P(\Sigma_i, v_{i-1}) \models p \\ &\text{iff} \quad Red(\Sigma_1^{i-1}, \Lambda_{i-1}) \cup Red(\Sigma_i, \Lambda_i) \models p \\ &\text{iff} \quad Red(\Sigma_1^i, \Lambda_i) \models p. \end{aligned}$$

The first equivalence is by the definition of v_i, the second and the third respectively by **A** and **B** below. The fourth equivalence is because $\Lambda_{i-1} \subseteq \Lambda_i$ and for each $L\phi \in Sf^L(\Sigma_1^{i-1})$ there is either $L\phi$ or $\neg L\phi$ in Λ_{i-1}. **A.** By the induction hypothesis $v_{i-1} \models p$ iff $Red(\Sigma_1^{i-1}, \Lambda_{i-1}) \models p$. We get the equivalence by Lemma 5.13 taking $\Delta = Red_P(\Sigma_i, v_{i-1}), \Gamma = v_{i-1}, \Sigma = Red(\Sigma_1^{i-1}, \Lambda_i)$. **B.** For $\phi, (\psi \rightarrow \phi) \in \Sigma_i$, $\phi \in Red_P(\Sigma_i, v_{i-1})$ iff for conjuncts $L\chi$ of ψ,

$$\begin{aligned} v_{i-1} \models_L \chi \quad &\text{iff} \quad Red(\Sigma_1^{i-1}, \Lambda_{i-1}) \models_L \chi && [31] \\ &\text{iff} \quad \Sigma_1^{i-1} \cup \Lambda_{i-1} \models_L \chi && [32] \\ &\text{iff} \quad L\chi \in \Lambda_i && [33] \end{aligned}$$

and similarly for conjuncts $\neg L\chi'$ of ψ, which by the definition of *Red* is equivalent to $\phi \in Red(\Sigma_i, \Lambda_i)$. Therefore $Red_P(\Sigma_i, v_{i-1}) = Red(\Sigma_i, \Lambda_i)$. Equivalence 31 above is by the induction hypothesis. Equivalence 32 is by Lemma 4.3. Equivalence 33 is by Lemma 4.2.

By Lemma 5.12 we get the induction step, i.e. for all χ in CF_{ae}, $v_i \models \chi$ iff $Red(\Sigma_1^i, \Lambda_i) \models \chi$. Thus $v_n \models_L \chi$ iff $Red(\Sigma_1^n, \Lambda_n) \models_L \chi$ iff (by Lemma 4.3) $\Sigma_1^n \cup \Lambda_n \models_L \chi$ iff (by Theorem 2.4) χ is in the unique stable expansion of Σ.

By Proposition 3.5 stratification can be computed in linear time, and using Assumption 3.4 the computation is $\mathcal{O}(|\Sigma|)$ time because by Proposition 5.11 computing $Red_P(\Sigma_i, v_{i-1})$ is linear in $|\Sigma_i|$ and this is done exactly once for each $\Sigma_i \subseteq \Sigma, 1 \leq i \leq n$. The size of $Red_P(\Sigma_i, v_{i-1})$ is smaller than or equal to that of Σ_i. Hence the respective computation with DG is linear in $|\Sigma_i|$. □

Theorem 5.15 *Let Σ be in SPC_{ae} and χ in CF_{ae}. The membership of χ in the unique stable expansion of Σ can be decided in $\mathcal{O}(n)$ time, where $n = |\Sigma| + |\chi|$.*

problem	variables	formulae	stratification (s)	consequences (s)
YSP	75	49	0.39	0.12
1	723	481	1.33	0.41
2	2181	1453	4.51	1.33
3	6555	4369	14.5	4.34

Figure 3. *Execution times for a number of examples*

Proof: By Lemma 5.14 the computation of the set v_n for $\Sigma = \Sigma_1^n$ is $\mathcal{O}(|\Sigma|)$ time. By Lemma 5.14 the membership in the unique stable expansion can be tested by $v_n \models_L \chi$, and this is $\mathcal{O}(|\chi|)$ time. $\square$

6. An implementation

We have implemented the general decision procedure for stratified propositional autoepistemic logic, as well as the decision procedures for the polynomial time classes presented in this article. All implementations are in Prolog. To get an idea about how practical the implementation is, we have tested it using a solution to the Yale Shooting Problem of Hanks and McDermott [HAN 87]. The axioms for the shooting scenario are the following.

holds(alive,s0)
∀S holds(loaded, result(load,S))
∀S holds(loaded, S) → holds(dead,result(shoot,S))
∀S holds(loaded, S) → ab(alive,shoot,S)
∀P∀A∀S ¬ab(P,A,S) ∧ holds(P,S) → holds(P,result(A,S))

We are interested in the sequence of events *load*, *wait*, and *shoot*, and the final situation *result(shoot,result(wait,result(load,s0)))*. Gelfond [GEL 88a] shows that autoepistemic logic gives the desired conclusions if the last formula of the problem statement is modified to

∀P∀A∀S ¬*L*ab(P,A,S) ∧ holds(P,S) → holds(P,result(A,S)).

The problem is stated in a quantificational version of autoepistemic logic, but we can remove the quantifiers and instantiate the free variables to obtain a set of formulae the atomic sentences of which can be treated like propositional variables. The instantiation process means substituting the relevant terms for the free variables occurring in the formulae. The relevant actions over which variables A vary are *load*, *wait*, and *shoot*, the relevant situations for variables S are *s0*, *result(load,s0)*, *result(wait,result(load,s0))*, and *result(shoot,result(wait,result(load,s0)))*, and the relevant facts are *loaded*, *dead*, and *alive*. The ground instantiation of the problem consists of 49 formulae in which 75 distinct propositional variables occur. See Figure 4.

1	h(alive, s0)
2	h(loaded, s0) → h(dead, r(s, s0)) ¬*L*ab(dead, s, s0) ∧ h(dead, s0) → h(dead, r(s, s0))
3	h(loaded, s0) → ab(alive, s, s0)
4	¬*L*ab(alive, s, s0) ∧ h(alive, s0) → h(alive, r(s, s0))
5	¬*L*ab(loaded, s, s0) ∧ h(loaded, s0) → h(loaded, r(s, s0))
6	¬*L*ab(dead, w, s0) ∧ h(dead, s0) → h(dead, r(w, s0))
7	¬*L*ab(alive, w, s0) ∧ h(alive, s0) → h(alive, r(w, s0))
8	¬*L*ab(loaded, w, s0) ∧ h(loaded, s0) → h(loaded, r(w, s0))
9	¬*L*ab(dead, l, s0) ∧ h(dead, s0) → h(dead, r(l, s0))
10	¬*L*ab(alive, l, s0) ∧ h(alive, s0) → h(alive, r(l, s0))
11	h(loaded, r(l, s0)) ¬*L*ab(loaded, l, s0) ∧ h(loaded, s0) → h(loaded, r(l, s0))
12	h(loaded, r(l, s0)) → h(dead, r(s, r(l, s0))) ¬*L*ab(dead, s, r(l, s0)) ∧ h(dead, r(l, s0)) → h(dead, r(s, r(l, s0)))
13	h(loaded, r(l, s0)) → ab(alive, s, r(l, s0))
14	¬*L*ab(alive, s, r(l, s0)) ∧ h(alive, r(l, s0)) → h(alive, r(s, r(l, s0)))
15	¬*L*ab(loaded, s, r(l, s0)) ∧ h(loaded, r(l, s0)) → h(loaded, r(s, r(l, s0)))
16	¬*L*ab(dead, w, r(l, s0)) ∧ h(dead, r(l, s0)) → h(dead, r(w, r(l, s0)))
17	¬*L*ab(alive, w, r(l, s0)) ∧ h(alive, r(l, s0)) → h(alive, r(w, r(l, s0)))
18	¬*L*ab(loaded, w, r(l, s0)) ∧ h(loaded, r(l, s0)) → h(loaded, r(w, r(l, s0)))
19	¬*L*ab(dead, l, r(l, s0)) ∧ h(dead, r(l, s0)) → h(dead, r(l, r(l, s0)))
20	¬*L*ab(alive, l, r(l, s0)) ∧ h(alive, r(l, s0)) → h(alive, r(l, r(l, s0)))
21	h(loaded, r(l, r(l, s0))) ¬*L*ab(loaded, l, r(l, s0)) ∧ h(loaded, r(l, s0)) → h(loaded, r(l, r(l, s0)))
22	h(loaded, r(w, r(l, s0))) → h(dead, r(s, r(w, r(l, s0)))) ¬*L*ab(dead, s, r(w, r(l, s0))) ∧ h(dead, r(w, r(l, s0))) → h(dead, r(s, r(w, r(l, s0))))
23	h(loaded, r(w, r(l, s0))) → ab(alive, s, r(w, r(l, s0)))
24	¬*L*ab(alive, s, r(w, r(l, s0))) ∧ h(alive, r(w, r(l, s0))) → h(alive, r(s, r(w, r(l, s0))))
25	¬*L*ab(loaded, s, r(w, r(l, s0))) ∧ h(loaded, r(w, r(l, s0))) → h(loaded, r(s, r(w, r(l, s0))))
26	¬*L*ab(dead, w, r(w, r(l, s0))) ∧ h(dead, r(w, r(l, s0))) → h(dead, r(w, r(w, r(l, s0))))
27	¬*L*ab(alive, w, r(w, r(l, s0))) ∧ h(alive, r(w, r(l, s0))) → h(alive, r(w, r(w, r(l, s0))))
28	¬*L*ab(loaded, w, r(w, r(l, s0))) ∧ h(loaded, r(w, r(l, s0))) → h(loaded, r(w, r(w, r(l, s0))))
29	¬*L*ab(dead, l, r(w, r(l, s0))) ∧ h(dead, r(w, r(l, s0))) → h(dead, r(l, r(w, r(l, s0))))
30	¬*L*ab(alive, l, r(w, r(l, s0))) ∧ h(alive, r(w, r(l, s0))) → h(alive, r(l, r(w, r(l, s0))))
31	h(loaded, r(l, r(w, r(l, s0)))) ¬*L*ab(loaded, l, r(w, r(l, s0))) ∧ h(loaded, r(w, r(l, s0))) → h(loaded, r(l, r(w, r(l, s0))))
32	h(loaded, r(s, r(w, r(l, s0)))) → h(dead, r(s, r(s, r(w, r(l, s0))))) ¬*L*ab(dead, s, r(s, r(w, r(l, s0)))) ∧ h(dead, r(s, r(w, r(l, s0)))) → h(dead, r(s, r(s, r(w, r(l, s0)))))
33	h(loaded, r(s, r(w, r(l, s0)))) → ab(alive, s, r(s, r(w, r(l, s0))))
34	¬*L*ab(alive, s, r(s, r(w, r(l, s0)))) ∧ h(alive, r(s, r(w, r(l, s0)))) → h(alive, r(s, r(s, r(w, r(l, s0)))))
35	¬*L*ab(loaded, s, r(s, r(w, r(l, s0)))) ∧ h(loaded, r(s, r(w, r(l, s0)))) → h(loaded, r(s, r(s, r(w, r(l, s0)))))
36	¬*L*ab(dead, w, r(s, r(w, r(l, s0)))) ∧ h(dead, r(s, r(w, r(l, s0)))) → h(dead, r(w, r(s, r(w, r(l, s0)))))
37	¬*L*ab(alive, w, r(s, r(w, r(l, s0)))) ∧ h(alive, r(s, r(w, r(l, s0)))) → h(alive, r(w, r(s, r(w, r(l, s0)))))
38	¬*L*ab(loaded, w, r(s, r(w, r(l, s0)))) ∧ h(loaded, r(s, r(w, r(l, s0)))) → h(loaded, r(w, r(s, r(w, r(l, s0)))))
39	¬*L*ab(dead, l, r(s, r(w, r(l, s0)))) ∧ h(dead, r(s, r(w, r(l, s0)))) → h(dead, r(l, r(s, r(w, r(l, s0)))))
40	¬*L*ab(alive, l, r(s, r(w, r(l, s0)))) ∧ h(alive, r(s, r(w, r(l, s0)))) → h(alive, r(l, r(s, r(w, r(l, s0)))))
41	h(loaded, r(l, r(s, r(w, r(l, s0))))) ¬*L*ab(loaded, l, r(s, r(w, r(l, s0)))) ∧ h(loaded, r(s, r(w, r(l, s0)))) → h(loaded, r(l, r(s, r(w, r(l, s0)))))

Figure 4. *A stratification of the Yale shooting problem*

Although the original set of formulae is not stratified because the predicates *ab* and *holds* are defined using each other through L, its ground instantiation is. Furthermore, the formulae are of the simple form allowed in the linear time class SPC_{ae}. The intuitive reason for the stratifiedness of the formulae is that the consequents of the implications describe the state of affairs in a later time than the antecedents, and because there are no cycles in the causal chains the formulae whose consequents describe earlier situations are lower in the stratification.

In Figure 3 execution times for four sample problems are shown. For each example we have measured the time taken by computing a stratification for the set of formulae, and the time taken by the computation of the propositional variables in the stable expansion of the formulae. All times are for program code compiled on Sun SPARC workstation in SICStus Prolog in native code compilation mode. The times have been measured using an internal facility of SICStus Prolog and are approximate. The first row identifies the problem, the second gives the number of distinct propositional variables in the problem, the third the number of formulae, the fourth the execution time of the algorithm computing a stratification for the set of formulae, and the fifth column gives the execution time of computing the propositional variables that belong to the unique stable expansion of the set of formulae.

The first row named YSP gives the runtimes of the solution to the Yale Shooting Problem described above. Rows 1,2,3 describe the execution of other variants obtained from the formula scheme of the Yale Shooting Problem. Instead of considering only the four relevant situations, all sequences of the actions *load*, *wait*, *shoot* of certain length are considered. This way the number of propositional variables and the number of formulae are significantly larger. The times for these modified problems are shown in Figure 3 as rows 1–3. The first example is for arbitrary sequences of three actions, the second for four actions, and the third for five actions.

The complexity analysis of Section 5 is based on a machine model with constant time array access. Because Prolog does not support such arrays, an array implementation based on balanced trees with $\log n$ average access time has been used instead. Hence the execution times do not grow linearly.

7. Applications

7.1. Nonmonotonic modal logics

McDermott and Doyle style modal nonmonotonic logics are closely related to autoepistemic logic. Marek et al. [MAR 93] show that for a wide range of modal nonmonotonic logics S, S-expansions of stratified sets coincide with stable expansions. Thus all the methods and results obtained for stratified autoepistemic logic are directly applicable to these logics.

Theorem 7.1 ([MAR 93]) *Let $\Sigma \subseteq \mathcal{L}_{ae}$ be stratified. Then for each logic S such that $\boldsymbol{N} \subseteq S$ and $S \subseteq \boldsymbol{KD45}$ or $S \subseteq \boldsymbol{SW5}$ the unique stable expansion of Σ is the unique S-expansion of Σ.*

7.2. Default logic

A notion of stratification can be introduced to Reiter's default logic in the same way as in autoepistemic logic, and similar complexity results hold in default logic as in autoepistemic logic. Konolige [KON 88] showed that under a certain translation, each extension of a default theory is the objective part of a stable expansion of a corresponding autoepistemic theory. In general there are stable expansions that do not correspond to extensions this way, but for stratified theories the correspondence is exact, as we will show below. The definition of extensions of default theories is based on a fixed point equation [REI 80] and for proofs it is more convenient to use a definition of extensions that has a more constructive flavour.

Theorem 7.2 ([REI 80]) *Let $E \subseteq \mathcal{L}$ be a set of well-formed formulae, and let $T = \langle D, W\rangle$ be a default theory. Define*

$$\begin{aligned} E_0 &= W \\ E_{i+1} &= \{\phi | E_i \models \phi\} \cup appl_{D,E}(E_i), i \geq 0 \end{aligned}$$

where $appl_{D,E}(E_i) = \{w| \ \dfrac{\alpha : \beta_1, \ldots, \beta_n}{w} \ \in D, \alpha \in E_i, \neg\beta_1, \ldots, \neg\beta_n \notin E\}$, and $\models$ is the classical logical consequence relation. Then E is an extension for T iff

$$E = \bigcup_{i=0}^{\infty} E_i.$$

The definition of stratification for autoepistemic logic (Definition 3.1) with the translation of a class of default theories to autoepistemic logic of Theorem 7.4 below, suggests a similar definition of stratification for default theories.

Definition 7.3 *A default theory $\langle D, W\rangle$ is stratified if*

1. *Formulae $\phi \in W$ are implications of the form $o(\phi) \rightarrow c(\phi)$ and the rules $d \in D$ are of the form $\alpha_d : \beta_{d1}, \ldots, \beta_{dm}/\gamma_d$.*
2. *The set $\{\gamma_d | d \in D\} \cup \{c(\phi)|\phi \in W\}$ is satisfiable.*
3. *There exists a set of indices $I = \{1, \ldots, n\}$ or $I = \{1, \ldots\}$ and a partition $\Sigma_i, i \in I$ of $\Sigma = D \cup W$ such that for all $j \in I$ the propositional variables occurring in $\{c(\phi)|\phi \in \Sigma_j\} \cup \{\gamma_d | d \in \Sigma_j\}$ do not occur in $\bigcup_{i=1}^{j-1} \Sigma_i$, or in a prerequisite or a justification of a default rule in Σ_j.*

Theorem 7.4 *Stratified default theories $\langle D, W\rangle$ can be translated into stratified autoepistemic theories by the following translation.*

$$tr_{dl}(D, W) = W \cup \{L\alpha \wedge \neg L\neg\beta_1 \wedge \cdots \wedge \neg L\neg\beta_n \rightarrow \gamma| \ \frac{\alpha : \beta_1, \ldots, \beta_n}{\gamma} \ \in D\}$$

The set of extensions of $\langle D, W\rangle$ is exactly the set of objective parts of the stable expansions of $tr_{dl}(D, W)$.

Proof: By Proposition 5.2 of [KON 88] every extension of $\langle D, W\rangle$ coincides with the objective part $E = \Delta \cap \mathcal{L}$ of some stable expansion Δ of $tr_{dl}(D, W)$. The proof of the opposite direction, i.e., that objective parts $E = \Delta \cap \mathcal{L}$ of stable expansions of $tr_{dl}(D, W)$ are extensions of $\langle D, W\rangle$, is by showing that $E = \bigcup_{i=0}^{\infty} E_i$ where the sets E_i are as in Theorem 7.2. First we prove the inclusion $\bigcup_{i=0}^{\infty} E_i \subseteq E$ by induction on i.

$(i = 0)$. $E_0 = W \subseteq E$ simply because $W \subseteq \Delta$. $(i \geq 1)$. Suppose $\gamma_d \in \mathrm{appl}_{D,E}(E_{i-1})$ for some d. Then $\alpha_d \in E_{i-1}$ and by the induction hypothesis $\alpha_d \in E \subseteq \Delta$. By the definition of stable expansions $L\alpha_d \in \Delta$. $\neg\beta_d \notin E$ and then $\neg L\neg\beta_d \in \Delta$ because Δ is a stable expansion. Because stable expansions are closed under logical consequence also $\gamma_d \in E$. Therefore $\bigcup_{i\geq 0} \mathrm{appl}_{D,E}(E_i) \subseteq E$ and $W \subseteq E$. Like $\bigcup_{i=0}^{\infty} E_i$ also E is closed under logical consequence and therefore $\bigcup_{i=0}^{\infty} E_i \subseteq E$.

Then we show the inclusion $E \subseteq \bigcup_{i=0}^{\infty} E_i$. This part of the proof rests on the stratifiedness of $\Sigma = tr_{dl}(D, W)$. Let $\Sigma_i, 1 \leq i \leq n$ be the strata of $\Sigma = tr_{dl}(D, W)$, and $\Lambda_i, 0 \leq i \leq n$ as in Theorem 4.1. By Theorem 2.4 and Lemma 4.3 the stable expansion of a stratified set Σ is $SE_\Sigma(\Lambda_n) = \{\phi | Red(\Sigma, \Lambda_n) \models_L \phi\}$. By the definition of $\models_L$ the objective part E of $SE_\Sigma(\Lambda_n)$ is $\{\phi | Red(\Sigma, \Lambda_n) \models \phi\}$. Because $\bigcup_{i=0}^{\infty} E_i$ is closed under logical consequence it suffices to show that $Red(\Sigma, \Lambda_n) \subseteq \bigcup_{i=0}^{\infty} E_i$. This is proved by induction on $i, 0 \leq i \leq n$.

$(i = 0)$. $Red(\Sigma_1^0, \Lambda_0) = \emptyset \subseteq \bigcup_{i=0}^{\infty} E_i$. $(i \geq 1)$. By definition $W \subseteq E$ is also contained in $\bigcup_{i=0}^{\infty} E_i$. Suppose that for $\gamma \in Red(\Sigma_1^i, \Lambda_i)$ there is a formula $(L\alpha \wedge \neg L\neg\beta_1 \wedge \cdots \wedge \neg L\neg\beta_s \rightarrow \gamma) \in \Sigma_i$. By Lemma 4.2 and the definitions of Red and Λ_i this means that for $L\alpha$, $Red(\Sigma_1^{i-1}, \Lambda_{i-1}) \models \alpha$ and for $\neg L\neg\beta_j, 1 \leq j \leq s$, $Red(\Sigma_1^{i-1}, \Lambda_{i-1}) \not\models \neg\beta_j$. By Lemma 4.2 $Red(\Sigma_1^n, \Lambda_n) \not\models \neg\beta_j$ and hence $\neg\beta_j \notin E$, and because of the inclusion $\bigcup_{i=0}^{\infty} E_i \subseteq E$ shown above finally $\neg\beta_j \notin \bigcup_{i=0}^{\infty} E_i$. Because $\neg\beta_j \notin \bigcup_{i=0}^{\infty}$ for $j \in \{1, \ldots, n\}$ and by the induction hypothesis $\alpha \in \bigcup_{i=0}^{\infty} E_i$, $\gamma \in \bigcup_{i=0}^{\infty} E_i$. □

Under the above translation stratified default theories map to stratified autoepistemic theories, and the counterparts of the quadratic and linear time classes of propositional autoepistemic logic in default logic are obtained. Define SHC_{dl} to be the class of default theories $\langle D, W\rangle$ where W is a set of Horn clauses, i.e., disjunctions of literals at most one of which is positive, and D a set of default rules of the form

$$\frac{\alpha : \beta_1, \ldots, \beta_n}{\gamma}$$

where α is a disjunction of conjunctions of literals with at most one negative literal in each conjunct, each β_i is a conjunction of disjunctions of literals with at most one positive literal in each disjunct, and γ is a literal. By noticing that the translation of Theorem 7.4 is linear time we get by Theorems 5.8 and 7.4 the following result ($|\Sigma|$ and $|\phi|$ denote of the sum of the lengths of the formulae involved).

Theorem 7.5 *For disjunctions of conjunctions of literals ϕ with at most one negative literal in each conjunct, the membership in the unique extension of a default theory T in SHC_{dl} can be decided in $\mathcal{O}(n^2)$ time, where $n = |T| + |\neg\phi|$.*

The class SPC_{dl} is like SHC_{dl} except that the formulae α are conjunctions of propositional variables, and β_j are disjunctions of negated propositional variables, and W is a set of program clauses, i.e. disjunctions of literals, exactly one of which is a positive literal. By Theorem 5.15 the following is immediate.

Theorem 7.6 *For conjunctions of propositional variables ϕ the membership in the unique extensions of default theories T in SPC_{dl} can be decided in $\mathcal{O}(n)$ time, where $n = |T| + |\phi|$.*

Kautz and Selman [KAU 91] and Stillman [STI 90] have analyzed the complexity of syntactically restricted classes of propositional default logic. In these classes of default theories the underlying classical reasoning is very easy, and the main result is that the restriction of the classical reasoning is not a sufficient condition for the tractability of default reasoning. As was pointed out in the beginning of Section 5, the classical reasoning is one of the two sources of complexity. After making also the second source – finding extensions - easy, we do reach polynomial time default reasoning. As expected, all intractable classes of simple default theories of [KAU 91, STI 90] become tractable if a stratification condition is imposed on them.

Because default rules in these classes are either normal or seminormal, default theories in these classes do not fulfill our stratification condition given earlier. Normality and semi-normality (with orderedness), i.e. the occurrence of the conclusion of a default rule in the justification, guarantee the existence of extensions. Also stratification is a sufficient condition – although a more restrictive one – for the existence of extensions. Thus the duplication of the consequent in the justification is unnecessary and in showing that the stratified counterparts of the simple default theories of [KAU 91, STI 90] are tractable, we just omit those parts of the justifications.

To show that in the stratified case *all* classes analyzed by [KAU 91, STI 90] are tractable, we show that the most general classes, i.e. the ones based on disjunction-free default rules and Horn clauses and the ones based on disjunction-free default rules and 2-literal clauses, are tractable. The size of a default theory $\langle D, W\rangle$ can be defined as the sum of the sizes of the formulae in W and in the prerequisites, justifications, and conclusions of the default rules in D.

Theorem 7.7 *Let ϕ be a disjunction of conjunctions of literals with at most one negative literal in each conjunct (resp. a disjunction of conjunctions of 2 literals). Deciding whether a formula ϕ belongs to the unique extension of a stratified disjunction-free default theory $\langle D, W\rangle$ where W is a set of Horn clauses (resp. 2-literal clauses) is solvable in $\mathcal{O}(n^2)$ time, where n is the sum of the sizes of the theory and ϕ.*

Proof: Theories with disjunction-free default rules are translated into autoepistemic logic using the translation

$$tr_{dlsn}\langle D, W\rangle = \{L\alpha_1 \wedge \cdots \wedge L\alpha_n \wedge \neg L\neg\beta \rightarrow \gamma | \frac{\alpha_1 \wedge \cdots \wedge \alpha_n : \beta \wedge \gamma}{\gamma} \in D\} \cup W$$

and we define that a disjunction-free default theory is stratified if the set of autoepistemic formulae obtained by the tr_{dlsn} translation is stratified. The translation differs in two respects from the Konolige translation given in Theorem 7.4. First, the translation ignores the normality part of the justification. Second, we have replaced $L(\alpha_1 \wedge \cdots \wedge \alpha_n)$ by the equivalent (in autoepistemic logic) formula $L\alpha_1 \wedge \cdots \wedge L\alpha_n$.

We show that in the computation of the algorithm of Theorem 4.5 all $\models_L$-consequence tests reduce to satisfiability testing of Horn-clauses when W is a set of Horn clauses, and to satisfiability testing of 2-literal clauses when W is a set of 2-literal clauses. Satisfiability testing in these cases is $\mathcal{O}(n)$ time [DOW 84, ASP 79]. The sets R_i (see Theorem 4.5) consist of formulae that are in W and formulae that are consequences of default rules in D. Formulae in W are Horn clauses (resp. 2-literal clauses) and the consequences of disjunction-free default rules are conjunctions of literals $\gamma_1 \wedge \cdots \wedge \gamma_n$ which are equivalent to sets of literals $\{\gamma_1, \ldots, \gamma_n\}$. Single literals are a special case of Horn clauses (resp. 2-literal clauses). Hence, the sets R_i consist of Horn clauses (resp. 2-literal clauses).

In the computation, the $\models_L$-consequence of formulae inside L from a set R_i is tested. These formulae are single literals α. By definition of $\models_L$, testing $R_i \models_L \alpha$ reduces to $R_i \models \alpha$, i.e., a classical logical consequence test of a literal α from a set of formulae R_i of classical propositional logic. This logical consequence test can be implemented as a satisfiability test of $R_i \cup \{\neg\alpha\}$, which is a set of Horn clauses (resp. 2-literal clauses).

Hence, by the complexity analysis given in Section 5 the computation of the full set associated with the disjunction-free default theory $\langle D, W\rangle$ is $\mathcal{O}(n^2)$ time, where n is the size of the default theory. The membership of formulae ϕ the negations of which are logically equivalent to sets of Horn clauses (resp. 2-literal clauses) in the unique extension can be tested in $\mathcal{O}(m)$ time, where m

is the sum of the sizes of $\langle D, W\rangle$ and ϕ. All this is $\mathcal{O}(m^2)$ time. $\square$

We conclude that as in stratified autoepistemic logic, also in stratified default logic the complexity of reasoning is determined by the complexity of the underlying classical reasoning. The stratified counterparts of the intractable restricted classes of default theories identified by [KAU 91, STI 90] are all tractable, and the possibility of further identification of such classes is immediate just like in autoepistemic logic.

7.3. Propositional logic programs

Gelfond and Lifschitz [GEL 88b] present a semantics for general logic programs called *the stable model semantics*. This semantics is essentially an autoepistemic one. Stratified logic programs translate into autoepistemic theories through the translation below, and the propositional variables in the unique stable expansion of the theory are exactly the ones in the unique stable model of the stratified program.

$$\begin{aligned} tr_{lp}(P) &= \{p_1 \wedge \cdots \wedge p_n \wedge \neg L q_1 \wedge \cdots \wedge \neg L q_m \rightarrow r | \\ &\quad (r \leftarrow p_1, \ldots, p_n, \text{not } q_1, \ldots, \text{not } q_m) \in P\} \end{aligned}$$

Stratified programs translate into sets of formulae in SPC_{ae}. Hence we immediately get the following result.

Theorem 7.8 *The unique stable model of stratified propositional logic programs P can be computed in time $\mathcal{O}(n)$, where $n = |P|$.*

Proof: By Theorem 3 of [GEL 88b] the unique stable model of P is the set of propositional variables in the stable expansion of $tr_{lp}(P)$. By our Theorem 5.14 this set can be computed in $\mathcal{O}(|P|)$ time. $\square$

This results improves Kautz and Selman's [KAU 91] $\mathcal{O}(n^3)$ result which is based on an algorithm that computes an extension for a restricted class of ordered default theories and a translation of stratified logic program into this class of theories. The algorithm is based on an ordering of literals given by the orderedness condition.

Our linear time result contrasts with the complexity of propositional general logic programs in the general case. For example, determining whether a propositional general logic program has a stable model is NP-complete [MAR 91a].

7.4. Truth-maintenance systems

Elkan [ELK 90] shows how sets of justifications of truth-maintenance systems of Doyle [DOY 79] can be seen as general propositional logic programs and also as a subclass of propositional autoepistemic theories. The result that if a set of justifications is stratified then the computation of the unique grounded model is linear time, is immediate.

Theorem 7.9 *The unique grounded model of a stratified set of justifications* Π *can be computed in* $\mathcal{O}(n)$ *time, where* $n = |\Pi|$.

Proof: By Theorem 4.9 of [ELK 90] M is a grounded model of Π iff M is the set of propositional variables in some stable expansion of the autoepistemic theory corresponding to Π. In the stratified case there is exactly one stable expansion, and by Lemma 5.14 the set can be computed in linear time in the size of the theory, i.e. the set of justifications. □

In the general case the problem of whether a set of justifications has a grounded model is NP-complete [URB 88].

8. Conclusions

The attempts to find subclasses of nonmonotonic reasoning which can be implemented efficiently by limiting the computational complexity of the required classical reasoning have produced very disappointing results [KAU 91, ELK 90, MAR 91a]. In this paper we follow a different approach. We identify the ability to "define" propositions using default assumptions about the same propositions as a significant source of computational complexity in nonmonotonic reasoning. The complexity of nonmonotonic reasoning can be reduced by disallowing such constructs, i.e., requiring the knowledge base to be stratified. We demonstrate this by developing an iterative (non-backtracking) algorithm for stratified autoepistemic theories the complexity of which is dominated by the required classical reasoning. Thus efficient subclasses of stratified nonmonotonic reasoning can be obtained by restricting the form of sentences in the knowledge base. As an example, we develop a quadratic and a linear time algorithm for limited subclasses of stratified autoepistemic theories. The results are shown to imply fast reasoning methods for stratified cases of default logic, logic programs, truth maintenance systems and nonmonotonic modal logics. The conclusion is that restriction to stratified knowledge bases gives a significant computational advantage and can reduce the computational complexity of nonmonotonic reasoning dramatically. For example, deciding whether a propositional logic program has a stable model is an NP-complete problem in the general case but for the stratified case we give a linear time algorithm which computes the unique stable model.

References

[AHO 74] Aho A. V., Hopcroft J. E., and Ullman J. D., *The design and analysis of computer algorithms*, Addison-Wesley, 1974.

[APT 88] Apt K. R., Blair H. A., and Walker A., Towards a theory of declarative knowledge, In Minker J., editor, *Foundations of Deductive Databases and Logic Programming*, pages 89–148, Morgan Kaufmann Publishers, Los Altos, CA, 1988.

[ASP 79] Aspvall B., Plass M. F., and Tarjan R. E., A linear time algorithm for testing the truth of certain quantified boolean formulas, *Information Processing Letters*, 8(3):121–123, 1979.

[BID 87] Bidoit N. and Froidevaux C., Minimalism subsumes default logic and circumscription in stratified logic programming, In *Proceedings of the Symposium on Logic in Computer Science*, pages 89–97, Ithaca, NY, June 1987, IEEE Computer Society Press.

[BID 91a] Bidoit N. and Froidevaux C., General logical databases and programs: Default logic semantics and stratification, *Information and Computation*, 91:15–54, 1991.

[BID 91b] Bidoit N. and Froidevaux C., Negation by default and unstratified logic programs, *Theoretical Computer Science*, 78:85–112, 1991.

[CHA 85] Chandra A. K. and Harel D., Horn clause queries and generalizations, *Journal of Logic Programming*, 2(1):1–15, 1985.

[CLA 78] Clark K. L., Negations as failure, In Gallaire H. and Minker J., editors, *Logic and Data Bases*, pages 293–322, Plenum Press, New York, 1978.

[DOW 84] Dowling W. F. and Gallier J. H., Linear-time algorithms for testing the satisfiability of propositional Horn formulae, *Journal of Logic Programming*, 1(3):267–284, 1984.

[DOY 79] Doyle J., A truth maintenance system, *Artificial Intelligence*, 12:231–272, 1979.

[ELK 90] Elkan C., A rational reconstruction of nonmonotonic truth maintenance systems, *Artificial Intelligence*, 43:219–234, 1990.

[FRO 88] Froidevaux C., More on stratified default theories, In *Proceedings of the 8th European Conference on Artificial Intelligence*, pages 492–494, München, August 1988, Pitman Publishing.

[FRO 92] Froidevaux C., Default logic for action rule-based systems, In Neumann B., editor, *Proceedings of the 10th European Conference on Artificial Intelligence*, pages 413–417, Vienna, 1992, John Wiley & Sons.

[GAL 88] Gallo G. and Scutellà M. G., Polynomially solvable satisfiability problems, *Information Processing Letters*, 29:221–226, November 1988.

[GAR 79] Garey M. R. and Johnson D. S., *Computers and Intractability*, W.H. Freeman and Company, San Francisco, 1979.

[GEL 87] Gelfond M., On stratified autoepistemic theories, In *Proceedings of the 6th National Conference on Artificial Intelligence*, pages 207–211, Seattle, July 1987, American Association for Artificial Intelligence.

[GEL 88a] Gelfond M., Autoepistemic logic and formalization of commonsense reasoning, preliminary report, In *Proceedings of the 2nd Workshop on Non-Monotonic Reasoning*, pages 176–186, Grassau, Germany, June 1988, Springer-Verlag.

[GEL 88b] Gelfond M. and Lifschitz V., The stable model semantics for logic programming, In *Proceedings of the 5th International Conference on Logic Programming*, pages 1070–1080, Seattle, August 1988, The MIT Press.

[GEL 90] Gelfond M. and Przymusinska H., Formalization of inheritance reasoning in autoepistemic logic, *Fundamenta Informaticae*, 13(4):430–443, 1990.

[GEL 91] Gelfond M., Lifschitz V., Przymusinska H., and Truszczyński M., Disjunctive defaults, In Allen J., Fikes R., and Sandewall E., editors, *Proceedings of the 2nd International Conference on Principles of Knowledge Representation and Reasoning*, pages 230–237, Cambridge, MA, April 1991, Morgan Kaufmann Publishers.

[GOT 92] Gottlob G., Complexity results for nonmonotonic logics, *Journal of Logic and Computation*, 2(3):397–425, June 1992.

[HAN 87] Hanks S. and McDermott D., Nonmonotonic logic and temporal projection, *Artificial Intelligence*, 33:379–412, 1987.

[KAK 90] Kakas A. C. and Mancarella P., Generalized stable models: a semantics for abduction, In Aiello L. C., editor, *Proceedings of the 9th European Conference on Artificial Intelligence*, pages 385–391, Stockholm, August 1990, Pitman Publishing.

[KAU 91] Kautz H. and Selman B., Hard problems for simple default theories, *Artificial Intelligence*, 49:243–279, 1991.

[KOL 91] Kolaitis P., The expressive power of stratified programs, *Information and Computation*, 90(1):50–66, January 1991.

[KON 88] Konolige K., On the relation between default and autoepistemic logic, *Artificial Intelligence*, 35:343–382, 1988.

[KON 89] Konolige K., On the relation between autoepistemic logic and circumscription, In *Proceedings of the 11th International Joint Conference on Artificial Intelligence*, pages 1213–1218, Detroit, August 1989, Morgan Kaufmann Publishers.

[LAS 87] Lassez C., McAloon K., and Port G., Stratification and knowledge base management, In *Proceedings of the 4th International Conference on Logic Programming*, volume 1, pages 136–151, 1987.

[LIF 88] LIFSCHITZ V., On the declarative semantics of logic programs with negation, In MINKER J., editor, *Foundations of Deductive Databases and Logic Programming*, pages 177–192, Morgan Kaufmann Publishers, Los Altos, 1988.

[MAR 89] MAREK W. and TRUSZCZYŃSKI M., Relating autoepistemic and default logics, In BRACHMAN R. J., LEVESQUE H. J., and REITER R., editors, *Proceedings of the 1st International Conference on Principles of Knowledge Representation and Reasoning*, pages 276–288, Toronto, May 1989.

[MAR 91a] MAREK W. and TRUSZCZYŃSKI M., Autoepistemic logic, *Journal of the ACM*, 38:588–619, 1991.

[MAR 91b] MAREK W. and TRUSZCZYŃSKI M., Computing intersection of autoepistemic expansions, In *Proceedings of the 1st International Workshop on Logic Programming and Non-monotonic Reasoning*, pages 37–50, Washington, DC, July 1991, The MIT Press.

[MAR 93] MAREK V. W., SCHWARZ G. F., and TRUSZCZYŃSKI M., Modal nonmonotonic logics: ranges, characterization, computation, *Journal of the ACM*, 40(4):963–990, September 1993.

[MCC 69] MCCARTHY J. and HAYES P. J., Some philosophical problems from the standpoint of artificial intelligence, In *Machine Intelligence 4*, pages 463–502, Edinburgh University Press, Edinburgh, 1969.

[MCC 77] MCCARTHY J., Epistemological problems of artificial intelligence, In *Proceedings of the 5th International Joint Conference on Artificial Intelligence*, pages 1038–1044, Cambridge, MA, August 1977.

[MIN 81] MINSKY M., A framework for representing knowledge, In HAUGELAND J., editor, *Mind Design*, pages 95–128, The MIT Press, Cambridge, MA, 1981.

[MOO 85] MOORE R. C., Semantical considerations on nonmonotonic logic, *Artificial Intelligence*, 25:75–94, 1985.

[NIE 90] NIEMELÄ I., Towards automatic autoepistemic reasoning, In VAN EIJCK J., editor, *Proceedings of the European Workshop on Logics in Artificial Intelligence—JELIA'90*, number 478 in Lecture Notes in Artificial Intelligence, pages 428–443, Amsterdam, September 1990, Springer-Verlag.

[PRZ 88] PRZYMUSINSKI T., On the declarative semantics of deductive databases and logic programs, In MINKER J., editor, *Foundations of Deductive Databases and Logic Programming*, pages 193–216, Morgan Kaufmann Publishers, Los Altos, 1988.

[REI 78] REITER R., On closed world data bases, In GALLAIRE H. and MINKER J., editors, *Logic and Data Bases*, pages 55–76, Plenum Press, New York, 1978.

[REI 80] REITER R., A logic for default reasoning, *Artificial Intelligence*, 13:81–132, 1980.

[REI 81] REITER R. and CRISCUOLO G., On interacting defaults, In *Proceedings of the 7th International Joint Conference on Artificial Intelligence*, pages 270–276, 1981.

[REI 87] REITER R., A theory of diagnosis from first principles, *Artificial Intelligence*, 32:57–95, 1987.

[SCH 78] SCHAEFER T. J., The complexity of satisfiability problems, In *Proceedings of the 10th Annual ACM Symposium on Theory of Computing*, pages 216–226, 1978.

[SHV 90] SHVARTS G., Autoepistemic modal logics, In *Proceedings of the 3rd Conference on Theoretical Aspects of Reasoning about Knowledge*, pages 97–109, Pacific Grove, CA, March 1990, Morgan Kaufmann Publishers.

[STI 90] STILLMAN J., It is not my default: the complexity of reasoning in default logic, In *Proceedings of the 9th National Conference on Artificial Intelligence*, pages 571–578, Boston, August 1990, Morgan Kaufmann Publishers.

[STI 92] STILLMAN J., The complexity of propositional default logics, In *Proceedings of the 10th National Conference on Artificial Intelligence*, pages 794–799, San Jose, CA, 1992.

[TAR 72] TARJAN R. E., Depth first search and linear graph algorithms, *SIAM Journal of Computing*, 1(2):146–160, 1972.

[URB 88] URBANSKI A., Formalizing non-monotonic truth maintenance systems, In O'SHEA T. and SGUREV V., editors, *Artificial Intelligence III: Methodology, Systems, Principles*, pages 43–50, Elsevier Science Publishers, 1988.

[VG88] VAN GELDER A., Negation as failure using tight derivations for general logic programs, In MINKER J., editor, *Foundations of Deductive Databases and Logic Programming*, pages 149–176, Morgan Kaufmann Publishers, Los Altos, CA, 1988.

[VG91] VAN GELDER A., ROSS K. A., and SCHLIPF J. S., The well-founded semantics for general logic programs, *Journal of the ACM*, 38(3):620–650, July 1991.

Ilkka Niemelä received his Master's degree in electrical engineering in 1987, his Licentiate's degree in computer science in 1989 and the degree of Doctor of Technology in computer science in 1993 all from Helsinki University of Technology. He is currently assistant professor in computer science at Helsinki University of Technology. Dr. Niemelä's research interests include knowledge representation, nonmonotonic reasoning, computational complexity, and automated theorem proving.

Jussi Rintanen received his Master's degree in computer science in 1992 and his Licentiate's degree in 1993, both from Helsinki University of Technology. He was awarded the annual M.Sc thesis award of the Finnish Society for Computer Science for his thesis the results of which are presented in this article. Currently he is a PhD student at Helsinki University of Technology. His research interests are priorities in nonmonotonic reasoning and belief revision.