On the Impact of Stratification on the Complexity of Nonmonotonic Reasoning

Ilkka Niemelä
Department of Computer Science
Helsinki University of Technology
Otakaari 1, SF–02150 ESPOO, Finland

Jussi Rintanen
Department of Computer Science
Helsinki University of Technology
Otakaari 1, SF–02150 ESPOO, Finland

Abstract

The ability to "define" propositions using default assumptions about the same propositions is identified as a source of additional computational complexity in nonmonotonic reasoning. If such constructs are not allowed, i.e. the knowledge base is stratified, 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 the knowledge base. As an example we derive quadratic and linear time algorithms for specific subclasses of stratified autoepistemic theories. The results are shown to imply efficient reasoning methods for stratified cases of default logic, logic programs, truth maintenance systems, and nonmonotonic modal logics.

1 INTRODUCTION

Nonmonotonic reasoning is an important aspect of many knowledge representation systems. 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 (Gottlob 1991). Furthermore, even very restricted subclasses of nonmonotonic reasoning where required classical reasoning can be done efficiently turn out to be computationally intractable. Examples of this are, e.g., simple cases of default logic (Kautz & Selman 1991), truth maintenance systems (Elkan 1990), and logic programs (Marek & Truszczyński 1991) which have NP-complete decision problems. 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 stratified knowledge bases. The notion of stratification has its origins in the logic programming community (Chandra & Harel 1985; Apt, Blair, & Walker 1988; Van Gelder 1988). A knowledge base is stratified if it can be partitioned into a set of levels (strata) such that on every level default assumptions are made only about propositions which have already been "defined" on lower levels. This restriction reduces expressivity at least in the sense that it rules out multiple extensions: a stratified knowledge base has exactly one possible set of correct conclusions. However, we show that the restriction 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 (Marek & Truszczyński 1991) 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) (Konolige 1988; Gelfond & Lifschitz 1988; Elkan 1990). Thus efficient decision methods developed for autoepistemic logic can be immediately applied to other forms of nonmonotonic reasoning. First we develop an iterative algorithm for reasoning in stratified autoepistemic theories. From this we derive a quadratic and a linear time algorithm for limited subclasses of stratified theories. The result are shown to imply fast reasoning algorithms for stratified cases of default logic, logic programs, truth maintenance systems, and nonmonotonic modal logics.

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 new $L\phi$ 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 (1985))**.** *Δ is a* stable expansion *of Σ iff*

$$\Delta = \{\phi \mid \Sigma \cup L\Delta \cup \neg L\overline{\Delta} \models \phi\} \qquad (1)$$

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

Stable expansions are infinite sets of formulae. A finitary characterization is needed for handling expansions computationally. Niemelä (1990) 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. $Sf^L(\phi)$ denotes the set of subformulae of the form $L\chi$ of ϕ. $Sf^{qL}(\phi)$ 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 $Sf^{qL}(\Sigma)$.

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 $L\phi \in Sf^L(\Sigma)$.*

For a set of premises Σ, the Σ-full sets are in a one-to-one correspondence with the stable expansions of Σ (Niemelä 1990). 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 (Niemelä 1990)):

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

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

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 $\Delta = \mathrm{SE}_\Sigma(\Lambda) = \{\phi \mid \Sigma \cup \Lambda \models_L \phi\}$ is the unique stable expansion of Σ such that $\Lambda \subseteq L\Delta \cup \neg L\overline{\Delta}$.*

Example 1. Let $\Sigma = \{Lp \to p, \neg Lp \to q\}$, where p and q are atomic. 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 $\Sigma \cup \Lambda_2 \not\models p$. So Σ has exactly two stable expansions $\mathrm{SE}_\Sigma(\{Lp\})$ and $\mathrm{SE}_\Sigma(\{\neg Lp\})$. $L\neg Lq$ belongs to the former but not to the latter because $\Sigma \cup \{Lp\} \models_L L\neg Lq$ but $\Sigma \cup \{\neg Lp\} \not\models_L L\neg Lq$. E.g., $\Sigma \cup \{Lp\} \models_L L\neg Lq$ can be verified as follows. As $\mathrm{SB}_{\Sigma\cup\{Lp\}}(q) = \emptyset$ and $\Sigma \cup \{Lp\} \not\models q$, $\Sigma \cup \{Lp\} \not\models_L q$. Thus $\mathrm{SB}_{\Sigma\cup\{Lp\}}(\neg Lq) = \{\neg Lq\}$. So $\Sigma \cup \{Lp\} \cup \mathrm{SB}_{\Sigma\cup\{Lp\}}(\neg Lq) \models \neg Lq$ which implies $\Sigma \cup \{Lp\} \models_L \neg Lq$. Hence $\mathrm{SB}_{\Sigma\cup\{Lp\}}(L\neg Lq) = \{L\neg Lq\}$ and thus $\Sigma \cup \{Lp\} \models_L L\neg Lq$. ■

3 STRATIFICATION

Marek and Truszczyński define a notion of a stratified set for propositional autoepistemic logic.

Definition 3.1 ((Marek & Truszczyński 1991))**.** *A set of formulae Σ is stratified if*

1. *the formulae $\phi \in \Sigma$ are of the form $a(\phi) \wedge o(\phi) \to c(\phi)$, where the subformulae $o(\phi)$ and $c(\phi)$ do not contain the operator L 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 Σ_1^{j-1} (where $\Sigma_a^b = \bigcup_{i=a}^b \Sigma_i$).*

If a set of premises is stratified, it has a unique stable expansion (Marek & Truszczyński 1991, Theorem 5.1). However, a non-stratified set can have several stable expansions. In fact there can be up to 2^n stable expansions for a set of premises with n $L\chi$ subformulae (Niemelä 1990). On the other hand, it is not necessary for a set to be stratified to have a unique stable expansion. The set $\{p, \neg Lp \to p\}$ is a simple example of this.

The notion of stratifiedness reduces expressivity because it rules out premises for which the agent has multiple competing sets of beliefs. When formalizing commonsense reasoning premises with multiple stable expansions seem to occur (e.g. (Gelfond 1988)). Kolaitis (1991) discusses the expressivity of stratified logic programs which are closely related to stratified autoepistemic theories. Stratification can be seen as an attempt to find a useful trade-off between the expressivity of a knowledge representation language and the computational complexity of the required reasoning effort. On one hand, stratification excludes multiple expansions and reduces expressivity and, on the other hand, it leads to efficient nonmonotonic reasoning methods as will be shown in this paper.

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 exponential time because of the satisfiability testing of $\{c(\phi)|\phi \in \Sigma\}$. In the polynomial time cases 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 $P_1, \dots, P_n$ of the propositional variables P 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 ((Marek & Truszczyński 1991)). *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 variables can easily be reduced to the computation of the strong components of a graph. 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 the 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 there is an occurrence of q in $a(\phi)$.

Theorem 3.3 ((Marek & Truszczyński 1991)). *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 2. The set

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

is not a-stratified and consequently it is not stratified. The characteristic graph has three strong components. The variables r and p 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 – because the edges between p and h are L-edges. ■

Example 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 component and there are no L-edges between them, s and r both occupy a singleton component. ■

For the tractable classes of autoepistemic reasoning 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 (1991) 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 (Lassez, McAloon, & Port 1987).

In the general case the size of the characteristic graph is quadratic on 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.

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

Proposition 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.*

Taking a set of formulae Σ as input and assigning a number for each propositional variable can be done in $\mathcal{O}(n \log v)$ time where n is the size of Σ and v is the number of distinct propositional variables in Σ.

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.*

In this restricted case Condition 2 of the definition of a stratified set can be tested in time $\mathcal{O}(|\Sigma|)$.

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 (1972) well-known algorithm for the strong components of a graph as presented in (Aho, Hopcroft, & Ullman 1974). Tarjan's algorithm runs in time $\mathcal{O}(n+e)$ 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 $\phi \in \Sigma_1$ that can be taken as the lowest stratum of a stratification Σ_1^n, and so on.

The size of the characteristic graph is linear on 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 set of 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 of 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 the same way as **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, Hopcroft, & Ullman 1974) 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] an array which is used for recognizing strong components during the traversal.

SEARCHC is called repeatedly for unmarked 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.

```
procedure SEARCHC(v);
begin
  mark v "old"
  dfnumber[v] := count;
  count := count + 1;
  lowlink[v] := dfnumber[v];
  push v on stack;
  for each vertex w on edges[v] do
    if w is marked "new" then
      begin
        SEARCHC(w);
        lowlink[v] := min(lowlink[v],lowlink[w]);
      end
    else
      if dfnumber[w] < dfnumber[v] and w is on stack
        then lowlink[v] := min(dfnumber[w],lowlink[v]);
    end if
  end for
  if lowlink[v] = dfnumber[v] then
    begin
      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 for
        concatenate formulae[x] to components[ccount];
      end while
    end
end
```

The first half of the algorithm traverses the characteristic graph depth-first and maintains the data structures for detection of the strong components. Like Tarjan's original algorithm it works in $\mathcal{O}(n+e)$ time where n and e are the number of nodes and edges, respectively. All our modifications are in the *if* statement that forms the second half of the algorithm.

First, for the formulae in the new component a new element is reserved in the array *components*. The *repeat* loop assigns each variable in the component the number of the component. Finally the *while* loop tests the component for the containment of an L-edge and concatenates the list of formulae belonging to the stratum to the array *components*. Constructing the lists of the array *components* is $\mathcal{O}(n)$ where n is the size of the set of formulae, 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. The tests for the containment of an edge through an L operator are within the $\mathcal{O}(n)$ bound as 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.

Example 4. Our algorithm computes for the set in Example 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 in 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

The next theorem gives an iterative algorithm for computing the Σ-full set of an arbitrary stratified set Σ. Because of the results of Marek and Truszczyński and on the other hand of Niemelä, the Σ-full set characterizes the unique stable expansion of Σ.

Theorem 4.1. *Let $\Sigma = \Sigma_1^n$ be a stratified set. Then $\Lambda = \Lambda_n$ defined by*
$\Lambda_0 = \emptyset$
$\Lambda_{i+1} = \Lambda_i \cup$
$\{L\chi | L\chi \in Sf^L(\Sigma_{i+1}) - Sf^L(\Sigma_1^i), \Sigma_1^i \cup \Lambda_i \models_L \chi\} \cup$
$\{\neg L\chi | L\chi \in Sf^L(\Sigma_{i+1}) - Sf^L(\Sigma_1^i), \Sigma_1^i \cup \Lambda_i \not\models_L \chi\}$
$(0 \leq i < n)$ is Σ-full and $SE_\Sigma(\Lambda)$ is the unique stable expansion of Σ.

For proving Theorem 4.1 the following lemma is essential.

Lemma 4.2. *Let $\Sigma = \Sigma_1^n$ be stratified and Λ_i 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 $\Sigma_1^j \cup \Lambda_j \models_x \chi$ where $\models_x$ is $\models$ or $\models_L, i < j \leq n$.*

Proof. By induction on i. When $i = 0, \Sigma_1^0 = \emptyset$ and $\Lambda_0 = \emptyset$. Otherwise as in the inductive case.

$(i \geq 0)$ We prove $\Sigma_1^i \cup \Lambda_i \models_L \chi$ iff $\Sigma_1^j \cup \Lambda_j \models_x \chi$ by induction on the L-depth s of χ. The L-depth of a formula is the maximum nesting level of L operators in it. The proof for the base case $s = 0$ is included in the proof of the inductive case as the only difference is that when $s = 0$ the sets $SB_{\Sigma_1^k \cup \Lambda_k}(\chi)$ for $k \in \{i, j\}$ are empty.

We have to show that for all $L\chi \in Sf^L(\Sigma_{i+1})$, $\Sigma_1^i \cup \Lambda_i \cup SB_{\Sigma_1^i \cup \Lambda_i}(\chi) \models \chi$ iff $\Sigma_1^j \cup \Lambda_j \models_x \chi, i < j \leq n$.

First note that **A.** $SB_{\Sigma_1^i \cup \Lambda_i}(\chi) = SB_{\Sigma_1^j \cup \Lambda_j}(\chi)$ by the induction hypothesis on s and the definition of $SB_\Sigma(\chi)$. **B.** $SB_{\Sigma_1^i \cup \Lambda_i}(\chi) \subseteq \Lambda_{i+1}$ (and furthermore $SB_{\Sigma_1^i \cup \Lambda_i}(\chi) \subseteq \Lambda_h$ for $h > i$) because for each $L\phi \in SB_{\Sigma_1^i \cup \Lambda_i}(\chi)$ there is $k \leq i$ such that $L\phi \in (Sf^L(\Sigma_{k+1}) - Sf^L(\Sigma_1^k))$ and by the induction hypothesis on i, $\Sigma_1^k \cup \Lambda_k \models_L \phi$, and by the equations of Theorem 4.1 $L\phi \in \Lambda_{k+1}$. Similarly for $\neg L\phi \in SB_{\Sigma_1^i \cup \Lambda_i}(\chi)$. By **A** and the monotonicity of $\models$ we get the implication $\Rightarrow$ for $\models_L$, and by monotonicity and **B** for $\models$.

$(\Leftarrow)$ We show that $\Sigma_1^i \cup \Lambda_i \not\models_L \chi$ implies $\Sigma_1^j \cup \Lambda_j \not\models_x \chi$. Let $\mathcal{M}$ be a model such that $\mathcal{M} \models \Sigma_1^i \cup \Lambda_i \cup SB_{\Sigma_1^i \cup \Lambda_i}(\chi)$ and $\mathcal{M} \not\models \chi$. Let $\mathcal{M}'$ be a model constructed by modifying $\mathcal{M}$ to satisfy $\Sigma_{i+1}^j \cup (\Lambda_j - \Lambda_i)$. This modification is possible without falsifying Σ_1^i because the set of propositional variables in $\{c(\phi) | \phi \in \Sigma_{i+1}^j\}$ is both disjoint from the propositional variables in $\Sigma_1^i \cup \{\chi\}$ and satisfiable according to the definition of stratification. Hence, $\Sigma_1^j \cup \Lambda_j \not\models \chi$. Although $SB_{\Sigma_1^i \cup \Lambda_i}(\chi)$ is not disjoint from Λ_j, it is contained in it because of **B** above. Then by **A** $SB_{\Sigma_1^j \cup \Lambda_j}(\chi) \subseteq \Lambda_j$ and therefore $\Sigma_1^j \cup \Lambda_j \not\models_L \chi$. □

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 $L\chi \in \Lambda$ iff there is $i < n$ for which $L\chi \in (Sf^L(\Sigma_{i+1}) - Sf^L(\Sigma_1^i))$ and $\Sigma_1^i \cup \Lambda_i \models_L \chi$ and by Lemma 4.2 $\Sigma_1^n \cup \Lambda \models \chi$. By Theorem 5.1 of (Marek & Truszczyński 1991) and Theorem 2.4 $SE_\Sigma(\Lambda)$ is the unique stable expansion of Σ. □

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 $\mathrm{Red}(\Sigma, \Lambda) = \{\psi \rightarrow \psi' | (\phi \wedge \psi \rightarrow \psi') \in \Sigma$ and 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 $\chi \in \mathcal{L}_{ae}$,*

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

Proof. Taking into account the equivalence $\phi \wedge \psi \rightarrow \psi' \equiv \phi \rightarrow (\psi \rightarrow \psi')$ we actually prove the lemma for $\mathrm{Red}(\Sigma, \Lambda) = \{\psi | (\phi \rightarrow \psi) \in \Sigma$ and the conjuncts of ϕ are in $\Lambda\}$, where ψ is written instead of $\psi \rightarrow \psi'$. The proof is by induction on the

L-depth s of χ. The base case $s = 0$ is included in the inductive case $s \geq 1$ as the only difference is that $SB_{\mathrm{Red}(\Sigma,\Lambda)}(\chi) = SB_{\Sigma\cup\Lambda}(\chi) = \emptyset$ when $s = 0$.

($\Rightarrow$) Suppose $\Sigma \cup \Lambda \not\models_L \chi$, i.e. there exists a model $\mathcal{M}$ such that $\mathcal{M} \models \Sigma \cup \Lambda \cup SB_{\Sigma\cup\Lambda}(\chi)$ and $\mathcal{M} \not\models \chi$. By the definition of SB and the induction hypothesis $SB_{\mathrm{Red}(\Sigma,\Lambda)}(\chi) = SB_{\Sigma\cup\Lambda}(\chi)$. Let $(\phi \rightarrow \psi) \in \Sigma$. If $\psi \in \mathrm{Red}(\Sigma,\Lambda)$, then the conjuncts of ϕ are in Λ and $\mathcal{M} \models \phi$. Now $\mathcal{M} \models \psi$ because $\mathcal{M} \models \phi \rightarrow \psi$. This shows that $\mathcal{M} \models \mathrm{Red}(\Sigma,\Lambda) \cup SB_{\mathrm{Red}(\Sigma,\Lambda)}(\chi)$ and $\mathrm{Red}(\Sigma,\Lambda) \not\models_L \chi$.

($\Leftarrow$) Suppose $\mathrm{Red}(\Sigma,\Lambda) \not\models_L \chi$, i.e. there exists a model $\mathcal{M}$ such that $\mathcal{M} \models \mathrm{Red}(\Sigma,\Lambda) \cup SB_{\mathrm{Red}(\Sigma,\Lambda)}(\chi)$ and $\mathcal{M} \not\models \chi$. By the definition of SB and the induction hypothesis $SB_{\mathrm{Red}(\Sigma,\Lambda)}(\chi) = SB_{\Sigma\cup\Lambda}(\chi)$, and $\mathcal{M} \models SB_{\Sigma\cup\Lambda}(\chi)$. Let $\mathcal{M}'$ be $\mathcal{M}$ modified to satisfy Λ. This modification is possible without affecting the truth values of $\mathrm{Red}(\Sigma,\Lambda)$, $SB_{\mathrm{Red}(\Sigma,\Lambda)}(\chi)$, and χ because $\mathrm{Red}(\Sigma,\Lambda) \subseteq \mathcal{L}$, for $L\phi \in (Sf^{qL}(\Lambda) \cap Sf^{qL}(SB_{\mathrm{Red}(\Sigma,\Lambda)}(\chi)))$, $L\phi \in \Lambda$ iff $L\phi \in SB_{\mathrm{Red}(\Sigma,\Lambda)}(\chi)$ by the definition of SB and our assumption (similarly with $\neg L\phi$), and because the truth value of no propositional variable in χ changes and the $L\phi$ subformulae of χ stay unchanged because the sub-beliefs of χ also do.

We still have to show that $\mathcal{M}' \models \Sigma$. For $(\phi \rightarrow \psi) \in \Sigma$ if all conjuncts of ϕ are in Λ then $\psi \in \mathrm{Red}(\Sigma,\Lambda)$ and therefore $\mathcal{M} \models \psi$, and further, $\mathcal{M}' \models \phi \rightarrow \psi$. 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 $\Sigma \cup \Lambda \not\models_L \chi$. □

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 $\mathrm{Red}(\Sigma,\Lambda) \subseteq \mathcal{L}$ which characterizes the stable expansion $SE_\Sigma(\Lambda)$.

Theorem 4.5. *Let $\Sigma = \Sigma_1^n$ be a stratified set. Let $R = R_n$ be defined by*

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

where $\mathrm{Red}_L(\Sigma,\Delta) = \{o(\phi) \rightarrow c(\phi) \mid \phi \in \Sigma, \Delta \models_L a(\phi)\}$. *Then $R = \mathrm{Red}(\Sigma,\Lambda)$, where Λ is the unique Σ-full set, and $\{\phi \mid R \models_L \phi\}$ is the unique stable expansion of Σ.*

Proof. We show by induction on i that for all $i = 0, \ldots, n$,

$$R_i = \mathrm{Red}(\Sigma_1^i, \Lambda_i). \qquad (2)$$

For $i = 0$, $R_i = \mathrm{Red}(\Sigma_1^i, \Lambda_i) = \emptyset$. ($\subseteq$) By the induction hypothesis $R_{i-1} = \mathrm{Red}(\Sigma_1^{i-1}, \Lambda_{i-1})$ and therefore $R_{i-1} \subseteq \mathrm{Red}(\Sigma_1^i, \Lambda_i)$. Let $o(\phi) \rightarrow c(\phi) \in \mathrm{Red}_L(\Sigma_i, R_{i-1})$. Thus $R_{i-1} \models_L a(\phi)$. By the induction hypothesis $\mathrm{Red}(\Sigma_1^{i-1}, \Lambda_{i-1}) \models_L a(\phi)$. The formula $a(\phi)$ is of the form $L\phi_1 \wedge \cdots \wedge L\phi_r \wedge \neg L\psi_1 \wedge \cdots \wedge \neg L\psi_s$ and by Lemma 4.4 for each $L\phi_j$, $\mathrm{Red}(\Sigma_1^{i-1}, \Lambda_{i-1}) \models_L \phi_j$ and for each $\neg L\psi_j$, $\mathrm{Red}(\Sigma_1^{i-1}, \Lambda_{i-1}) \not\models_L \psi_j$. By Lemma 4.3 $\Sigma_1^{i-1} \cup \Lambda_{i-1} \models_L \phi_j$ and $\Sigma_1^{i-1} \cup \Lambda_{i-1} \not\models_L \psi_j$. For each $L\phi_j$ there is some $k \leq i$ such that $L\phi_j \in Sf^L(\Sigma_k) - Sf^L(\Sigma_1^{k-1})$. By Lemma 4.2 $\Sigma_1^{k-1} \cup \Lambda_{k-1} \models_L \phi_j$ and thus $L\phi_j \in \Lambda_k \subseteq \Lambda_i$. Similarly it can be shown that $\neg L\psi_j \in \Lambda_i$. Thus $o(\phi) \rightarrow c(\phi) \in \mathrm{Red}(\Sigma_1^i, \Lambda_i)$ and $R_i \subseteq \mathrm{Red}(\Sigma_1^i, \Lambda_i)$.

($\supseteq$) Let $o(\phi) \rightarrow c(\phi) \in \mathrm{Red}(\Sigma_1^i, \Lambda_i)$. If $\phi \in \Sigma_1^{i-1}$, then $o(\phi) \rightarrow c(\phi) \in \mathrm{Red}(\Sigma_1^{i-1}, \Lambda_i) = \mathrm{Red}(\Sigma_1^{i-1}, \Lambda_{i-1})$. Therefore by the induction hypothesis $o(\phi) \rightarrow c(\phi) \in R_{i-1}$. Let $\phi \in \Sigma_i$. The conjuncts in $a(\phi)$ are in Λ_i where $a(\phi)$ is of the form $L\phi_1 \wedge \cdots \wedge L\phi_r \wedge \neg L\psi_1 \wedge \cdots \wedge \neg L\psi_s$. By Lemma 4.2 $\Sigma_1^{i-1} \cup \Lambda_{i-1} \models_L \phi_j$ and $\Sigma_1^{i-1} \cup \Lambda_{i-1} \not\models_L \psi_j$ for each $L\phi_j, \neg L\psi_j$. By Lemma 4.3 $\mathrm{Red}(\Sigma_1^{i-1}, \Lambda_{i-1}) \models_L \phi_j$ and $\mathrm{Red}(\Sigma_1^{i-1}, \Lambda_{i-1}) \not\models_L \psi_j$ and by the induction hypothesis $R_{i-1} \models_L a(\phi)$. Thus $o(\phi) \rightarrow c(\phi) \in \mathrm{Red}_L(\Sigma_i, R_{i-1}) \subseteq R_i$. Hence $\mathrm{Red}(\Sigma_1^i, \Lambda_i) \subseteq R_i$.

We have shown that $R_n = \mathrm{Red}(\Sigma_1^n, \Lambda_n)$. Thus $R_n \models_L \phi$ iff $\mathrm{Red}(\Sigma_1^n, \Lambda_n) \models_L \phi$ iff $\Sigma_1^n \cup \Lambda_n \models_L \phi$ by Lemma 4.3. Thus by Theorem 4.1 $\{\phi \mid R_n \models_L \phi\}$ is the unique stable expansion of Σ. □

Example 5. The table below demonstrates the computation of Λ by the algorithm in Theorem 4.1, and the computation of $R = \mathrm{Red}(\Sigma,\Lambda)$ with 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

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.

As a basis of analyzing the complexity of computing the stable expansions of stratified sets we use the algorithm in Theorem 4.5 for computing 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 appear.

Lemma 5.1. *Let $\Sigma \subseteq \mathcal{L}$ and $\chi(L\phi)$ 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 $\top$. 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 both 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 $\bot$. □

Lemma 5.2. *$F(\chi, \Sigma)$ in Equation 3 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.*

$$F(\chi, \Sigma) = R(|\chi| + |\Sigma| - \sum_{L\phi \in Sf^{qL}(\chi)} |\phi|) + \sum_{L\phi \in Sf^{qL}(\chi)} F(\phi, \Sigma) \qquad (3)$$

and the equation with recursion removed is

$$F(\chi, \Sigma) = R(|\chi| + |\Sigma| - \sum_{L\phi \in Sf^{qL}(\chi)} |\phi|) + \sum_{L\phi \in Sf^{L}(\chi)} R(|\phi| + |\Sigma| - \sum_{L\psi \in Sf^{qL}(\phi)} |\psi|) \qquad (4)$$

Proof. The second summand of the rhs of Equation 3 corresponds to the computation of the members of $SB_\Sigma(\chi)$ using $\models_L$. The first summand corresponds to the consequence test $\Sigma \models_L \chi$, i.e. the test of satisfiability of $\{\neg\chi\} \cup \Sigma \cup SB_\Sigma(\chi)$. 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 $\bot$ 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 $m_i = |o(\phi_i) \to c(\phi_i)|$. 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:

$$\sum_{i=1}^{r} (n_i + m_i + 1) \qquad (5)$$

and an upper bound for the resources needed for the consequence tests is

$$\sum_{i=1}^{r} F(a(\phi_i), \bigcup_{j=1}^{i-1} \{o(\phi_j) \to c(\phi_j)\}) \qquad (6)$$

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. For the members of this class the $\models$ consequence can be tested by using the linear time algorithm of Dowling and Gallier (1984) and for SHC$_{ae}$ the algorithm of the previous section runs in polynomial time.

Definition 5.3. *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 5.4. *SHC_{ae} is the class of finite stratified sets of formulae ϕ of the form $a(\phi) \wedge o(\phi) \to c(\phi)$ 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 the syntactic form of the formulae in sets in SHC$_{ae}$.

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

Theorem 5.5. *For a set Σ in SHC_{ae} the set* Red(Σ, Λ), *where Λ is the unique Σ-full set, can be computed in time $\mathcal{O}(n^2)$ where $n = |\Sigma|$.*

Proof. By Proposition 3.5 a stratification can be determined in $\mathcal{O}(|\Sigma|)$ time, and the rest of the computation of the algorithm in Theorem 4.5 is clearly dominated by the consequence tests. All consequence tests reduce to satisfiability tests for formulae that are in conjunctive normal form with at most one positive literal in each conjunct, so that the linear time satisfiability algorithm of (Dowling & Gallier 1984) 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 5.2:

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

By reordering the terms:

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

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 6 (by the definition of $\mathcal{O}$ the constant factor $a + c$ appearing in each term can be left out).

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

Thus, we obtain the $\mathcal{O}(n^2)$ upper bound for the logical consequence tests. □

Theorem 5.6. *Given* Red(Σ, Λ) *where* $\Sigma \in SHC_{ae}$ *and* Λ *is a* Σ*-full set, the membership in the unique stable expansion of* Σ $\{\phi|\text{Red}(\Sigma, \Lambda) \models_L \phi\}$ *for formulae* ϕ *in* HF_{ae} *can be decided in time* $\mathcal{O}(n^2)$*, where* $n = |\Sigma| + |\neg\phi|$*.*

Proof. By the second equation of Lemma 5.2 the amount of resources needed is bounded by $|\Sigma| \cdot |\neg\phi| + |\neg\phi|$ as shown in the proof of Theorem 5.5, and this is $\mathcal{O}(n^2)$. □

Theorem 5.7. *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 time* $\mathcal{O}(n^2)$ *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 (Schaefer 1978; Gallo & Scutellà 1988). 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 5.8. *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 5.9. SPC_{ae} *is the class of finite stratified sets of formulae* ϕ *of the form* $a(\phi) \wedge o(\phi) \to c(\phi)$ *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$_{ae}$:

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

In (Dowling & Gallier 1984) two linear time algorithms are given for testing the satisfiability of a set of Horn clauses. We modify the first 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 (Dowling & Gallier 1984). 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 $a_1 \wedge \cdots \wedge a_n \to b$ 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 the Assumption 3.4 is essential.

Proposition 5.10. *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 time $\mathcal{O}(|\Sigma|)$*.*

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 following procedure.

```
 function DG($\Sigma$, 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 $\neq$ 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
```

Under Assumption 3.4 the computation is linear time on the size of Σ. The amount of computation inside the while loop is proportional to the number of negative literals in Σ. □

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

$v \models_L \chi$ iff

$$\text{for each conjunct } \phi \text{ of } \chi \left\{ \begin{array}{l} \text{if } \phi \text{ is atomic then } \phi \in v \\ \text{if } \phi = L\psi \text{ then } v \models_L \psi \\ \text{if } \phi = \neg L\psi \text{ then } v \not\models_L \psi \end{array} \right.$$

The following two lemmata are needed for establishing Lemma 5.13 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 5.11. *Let χ be in CF_{ae}, $\Sigma \subseteq \mathcal{L}$, and $v = \{p | p$ is a propositional variable$, \Sigma \models p\}$. Then $v \models_L \chi$ iff $\Sigma \models_L \chi$.*

Proof. By induction on the L-depth s of χ. Let $s = 0$. 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 $v \models_L \chi$. Let $s \geq 1$. 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 can be shown as in the case $s = 0$. □

Lemma 5.12. *Let Σ be a set of program clauses and $\Gamma = \{p | p$ is a propositional variable$, \Sigma \models p\}$. 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. 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 $\Sigma \cup \Delta \not\models p$. □

Lemma 5.13. *For $\Sigma = \Sigma_1^n$ in SPC_{ae} and formulae 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*

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

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

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

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

The first equivalence is by the definition of v_i, the second and the third by **A** and **B** below, respectively. **A.** By the induction hypothesis $v_{i-1} \models p$ iff $\mathrm{Red}(\Sigma_1^{i-1}, \Lambda_{i-1}) \models p$. We get the equivalence by Lemma 5.12 taking $\Delta = \mathrm{Red}_L(\Sigma_i, v_{i-1}), \Gamma = v_{i-1}, \Sigma = \mathrm{Red}(\Sigma_1^{i-1}, \Lambda_i)$. **B.** By the definitions of Red and Red_L a formula ϕ is in $\mathrm{Red}_L(\Sigma_i, v_{i-1})$ if $(\psi \rightarrow \phi) \in \Sigma_i$ and for the conjuncts $L\chi, \neg L\chi'$ of ψ, $v_{i-1} \models_L \chi$ and $v_{i-1} \not\models_L \chi'$, and in $\mathrm{Red}(\Sigma_i, \Lambda_i)$ if $L\chi, \neg L\chi' \in \Lambda_i$. By Lemmata 4.2 and 4.3 $\mathrm{Red}(\Sigma_1^{i-1}, \Lambda_{i-1}) \models_L \chi$ and $\mathrm{Red}(\Sigma_1^{i-1}, \Lambda_{i-1}) \not\models_L \chi'$. By the induction hypothesis $v_{i-1} \models_L \chi$ iff $\mathrm{Red}(\Sigma_1^{i-1}, \Lambda_{i-1}) \models_L \chi$. Therefore $\mathrm{Red}_L(\Sigma_i, v_{i-1}) = \mathrm{Red}(\Sigma_i, \Lambda_i)$.

By Lemma 5.11 we get the induction step, i.e. for all χ in CF_{ae}, $v_i \models \chi$ iff $\mathrm{Red}(\Sigma_1^i, \Lambda_i) \models \chi$.

By Theorem 2.4 and Lemma 4.3 $\chi \in \mathrm{CF}_{ae}$ is in the unique stable expansion of Σ iff $v_i \models_L \chi$.

By Proposition 3.5 stratification can be computed in linear time, and using Assumption 3.4 the computation is $\mathcal{O}(|\Sigma|)$ time because computing $\mathrm{Red}_L(\Sigma_i, v_{i-1})$ 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 $\mathrm{Red}_L(\Sigma_i, v_{i-1})$ is smaller than or equal to that of Σ_i and therefore the respective computation with DG is linear in $|\Sigma_i|$. □

Theorem 5.14. *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 time $\mathcal{O}(n)$, where $n = |\Sigma| + |\chi|$.*

Proof. By Lemma 5.13 the computation of the set v_n for $\Sigma = \Sigma_1^n$ is $\mathcal{O}(|\Sigma|)$. The membership in the unique stable expansion can be tested by $v_n \models_L \chi$, and this is $\mathcal{O}(|\chi|)$. □

6 APPLICATIONS

Autoepistemic logic is closely related to McDermott and Doyle style nonmonotonic modal logics. Marek et al. (1991) show that for a wide range of modal logics S, S-expansions coincide with stable expansions in the stratified case. Thus all the methods and results obtained for stratified autoepistemic logic are directly applicable to these logics.

Theorem 6.1 (Marek et al. (1991)). *Let $\Sigma \subseteq \mathcal{L}_{ae}$ be stratified. Then for each logic S such that* **N** $\subseteq S$ *and* $S \subseteq$ **KD45** *or* $S \subseteq$ **S4** *the unique stable expansion of Σ is the unique S-expansion of Σ.*

Reiter's (1980) default logic is one of the leading formalizations of nonmonotonic reasoning. Konolige (1988) shows that under a suitable translation of a default theory extensions of the theory correspond to stable expansions of the translated theory satisfying a special groundedness condition. In the case of stratified default theories the special groundedness condition is not required. So the iterative algorithm in Theorem 4.5 yields efficient methods for computing extensions of stratified default theories based on subsets of propositional logic for which efficient satisfiability tests are available. As a straightforward example of this we show that the linear time algorithm in Lemma 5.13 implies a linear time algorithm for a similar class of stratified default logic.

We call (D, W) a stratified Horn default theory if

1. W is a finite set of propositional program clauses and
2. D is a finite set of default rules $\frac{\alpha:\beta_1,\ldots,\beta_n}{\gamma}$ where the prerequisite α is a conjunction of propositional variables and each justification β_i is a disjunction of negated propositional variables and the conclusion γ is a propositional variable and there is a partition $D_1, \ldots, D_n$ of D such that if γ is a conclusion of a default rule in D_i, then γ does not appear in $D_1 \cup \ldots \cup D_{i-1}$, in negative literals in W nor in any prerequisite or justification in D_i.

Lemma 6.2. *Let (D, W) be a stratified Horn default theory. Then $E = \Delta \cap \mathcal{L}$ is the unique extension of (D, W) where Δ is the unique stable expansion of the stratified autoepistemic theory*

$$\begin{aligned} W \quad \cup \quad & \{L\alpha \wedge \neg L\neg\beta_1 \wedge \ldots \wedge \neg L\neg\beta_n \rightarrow \gamma \mid \\ & \frac{\alpha : \beta_1, \ldots, \beta_n}{\gamma} \in D\}. \end{aligned} \tag{7}$$

Theorem 6.3. *Let (D, W) be a stratified Horn default theory. Then the atomic part of the unique extension of (D, W) can be computed in linear time.*

Stable model semantics (Gelfond & Lifschitz 1988) and well-founded semantics (Van Gelder, Ross, & Schlipf 1991) are the leading declarative semantics for general logic programs. For stratified propositional logic programs the stable model semantics and the well-founded semantics coincide. Logic programs can be seen as sets of autoepistemic formulae, e.g. using the following translation proposed by Gelfond and Lifschitz (1988).

$$\begin{aligned} & \mathrm{tr}_{LP}(a_0 \leftarrow a_1, \ldots, a_m, \mathrm{not}\, a_{m+1}, \ldots, \mathrm{not}\, a_n) = \\ & \quad (a_1 \wedge \ldots \wedge a_m \wedge \neg La_{m+1} \wedge \ldots \wedge \neg La_n) \rightarrow a_0 \end{aligned} \tag{8}$$

When using this translation stable models and stable expansions are closely related in the stratified case. The correspondence is described in the following lemma which a direct consequence of the results of Gelfond and Lifschitz (1988).

Lemma 6.4. *Let P be a propositional stratified general logic program. $S = \Delta \cap \mathrm{At}$ is the unique stable model (well-founded model) of P where Δ is the unique stable expansion of $\mathrm{tr}_{LP}(P)$ and At is the set of propositional variables.*

Deciding whether a propositional logic program has a stable model is NP-complete in the general case (Marek & Truszczyński 1991). Kautz and Selman (1991) propose a cubic algorithm for computing the stable model of a stratified logic program using a translation to default logic. This result can be improved using the algorithm in Lemma 5.13.

Theorem 6.5. *Let P be a stratified general propositional logic program. The unique stable model (well-founded model) of P can be computed in linear time.*

Elkan (1990) shows that the justifications in a nonmonotonic truth maintenance system (TMS) can be seen as a propositional general logic program and that the grounded model computed by the TMS is a stable model of the corresponding logic program.

Corollary 6.6. *Let J be a stratified set of justifications. Then the unique grounded model of J can be computed in linear time.*

7 CONCLUSIONS

The attempts to find subclasses of nonmonotonic reasoning which can be implemented efficiently by limiting the computational complexity of required classical reasoning have produced very disappointing results (Kautz & Selman 1991; Elkan 1990; Marek & Truszczyński 1991). In this paper we follow a different approach. We identify the ability to "define" propositions using default assumptions about the same propositions as a source of additional computational complexity in nonmonotonic reasoning. Disallowing such constructs, i.e. requiring the knowledge base to be stratified, gives a significant computational advantage. We demonstrate this by developing an iterative (non-backtracking) algorithm for stratified autoepistemic theories the complexity of which is dominated by 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. E.g., 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.

Acknowledgements Ilkka Niemelä gratefully acknowledges the support from the following foundations: Foundation of Technology, Jenny and Antti Wihuri Foundation, Emil Aaltonen Foundation, Alfred Kordelin Foundation, as well as Heikki and Hilma Honkanen Foundation.

References

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

Apt, K. R.; Blair, H. A.; and Walker, A. 1988. Towards a theory of declarative knowledge. In Minker, J., ed., *Foundations of Deductive Databases and Logic Programming.* Los Altos, California: Morgan Kaufmann Publishers. 89–148.

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

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

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

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

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

Gelfond, M. 1988. Autoepistemic logic and formalization of commonsense reasoning, preliminary report. In *Proceedings of the 2nd Workshop on Non-Monotonic Reasoning*, number 346 in Lecture Notes in Artificial Intelligence, 176–186. Grassau, Germany: Springer-Verlag.

Gottlob, G. 1991. Complexity results for nonmonotonic logics. Technical Report CD-TR 91/24, Christian Doppler Laboratory for Expert Systems, Institut für Informationssysteme, Technische Universität Wien, Vienna.

Kautz, H., and Selman, B. 1991. Hard problems for simple default theories. *Artificial Intelligence* 49(1-3):243–279.

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

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

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

Marek, W., and Truszczyński, M. 1991. Autoepistemic logic. *Journal of the ACM* 38:588–619.

Marek, W.; Shvarts, G. F.; and Truszczyński, M. 1991. Modal nonmonotonic logics: Ranges, characterization, computation. In Allen, J.; Fikes, R.; and Sandewall, E., eds., *Principles of Knowledge Representation and Reasoning: Proceedings of the Second International Conference (KR '91)*, 395–404. Cambridge, Massachusetts: Morgan Kaufmann Publishers.

Moore, R. C. 1985. Semantical considerations on nonmonotonic logic. *Artificial Intelligence* 25(1):75–94.

Niemelä, I. 1990. Towards automatic autoepistemic reasoning. In Van Eijck, J., ed., *Proceedings of the European Workshop on Logics in Artificial Intelligence—JELIA'90*, number 478 in Lecture Notes in Artificial Intelligence, 428–443. Amsterdam: Springer-Verlag.

Reiter, R. 1980. A logic for default reasoning. *Artificial Intelligence* 13(1-2):81–132.

Schaefer, T. J. 1978. The complexity of satisfiability problems. In *Proceedings of the 10th Annual ACM Symposium on Theory of Computing*, 216–226.

Tarjan, R. E. 1972. Depth first search and linear graph algorithms. *SIAM Journal on Computing* 1(2):146–160.

Van Gelder, A.; Ross, K. A.; and Schlipf, J. S. 1991. The well-founded semantics for general logic programs. *Journal of the ACM* 38(3):620–650.

Van Gelder, A. 1988. Negation as failure using tight derivations for general logic programs. In Minker, J., ed., *Foundations of Deductive Databases and Logic Programming.* Los Altos, California: Morgan Kaufmann Publishers. 149–176.