ELSEVIER

Artificial Intelligence 74 (1995) 249-310

Artificial Intelligence

Tractable reasoning via approximation ★

Marco Schaerf [a,b,1], Marco Cadoli [b,*]

[a] *Istituto di Elettrotecnica, Università di Cagliari, Piazza d'Armi, 90123 Cagliari, Italy*

[b] *Dipartimento di Informatica e Sistemistica, Università di Roma "La Sapienza", via Salaria 113, 00198 Roma, Italy*

Received May 1993; revised November 1993

Abstract

Problems in logic are well known to be hard to solve in the worst case. Two different strategies for dealing with this aspect are known from the literature: language restriction and theory approximation.

In this paper we are concerned with the second strategy. Our main goal is to define a semantically well-founded logic for approximate reasoning, which is justifiable from the intuitive point of view, and to provide fast algorithms for dealing with it even when using expressive languages. We also want our logic to be useful to perform approximate reasoning in different contexts. We define a method for the approximation of decision reasoning problems based on multivalued logics. Our work expands and generalizes, in several directions, ideas presented by other researchers. The major features of our technique are: (1) approximate answers give semantically clear information about the problem at hand; (2) approximate answers are easier to compute than answers to the original problem; (3) approximate answers can be improved, and eventually they converge to the right answer; (4) both sound approximations and complete ones are described.

The method we propose is flexible enough to be applied to a wide range of reasoning problems. In our research we considered approximation of several decidable problems with different worst-case complexity, involving both propositional and first-order languages. In particular we defined approximation techniques for: propositional logic, fragments of first-order logic (concept descrip-

★ Work supported by the ESPRIT Basic Research Action 6810-COMPULOG II and by the Progetto Finalizzato Sistemi Informatici e Calcolo Parallelo of the CNR (Italian Research Council). Parts of this paper appeared in preliminary form in papers presented at the Fourth Conference on Theoretical Aspects of Reasoning about Knowledge (TARK-92), the Third International Conference on the Principles of Knowledge Representation and Reasoning (KR-92) and the Second Italian Conference on AI (AI*IA-91).

* Corresponding author. E-mail: cadoli@assi.dis.uniroma1.it.

[1] E-mail: schaerf@assi.dis.uniroma1.it.

SSDI 0004-3702(94)00009-P

tion languages) and modal logic. In our research we also addressed the issue of representing the knowledge of a reasoner with limited resources and how to use such a knowledge for approximate reasoning purposes.

1. Introduction

The benefits of formalizing AI problems in logic are manifold [45,60]. Nevertheless we have to pay a price for that: one of the major drawbacks of logic as an AI tool is in that if we formalize AI problems as logical problems then typically the tasks that we are supposed to perform have high computational complexity. Problems in logic are well known to be hard to solve in the worst case: the prototypical NP-complete problem is to check whether a formula of propositional logic is consistent [17], while the prototypical r.e.-complete problem is to check whether a formula of first-order logic is inconsistent [64]. We notice that—from the AI perspective—consistency checking is a basic task, which is subsumed by many formalisms in knowledge representation (e.g. knowledge update.)

The computational drawback caused by logical formalization of problems is not a peculiarity of AI: Vardi in [73] shows how an increase in complexity characterizes the formalization of databases as *logical* objects versus their characterization as *physical* objects.

In fact the use of "fancy" forms of logic as tools for knowledge representation makes the computational drawback even more sensible. Logical formalisms that are used in AI typically have higher computational complexity than classical—propositional or first-order—logic. As an example, consistency checking in propositional classical logic is NP-complete, while consistency checking in the propositional modal logic $S4$—a logic which is commonly used for representing the *knowledge* modality [43]—is PSPACE-complete [52].

Since researchers realized this fundamental drawback, two different strategies emerged in the literature:

(1) According to the ideas presented in [6], we can restrict the language used for representing knowledge so that the formalization of interesting cases is still possible, but resulting tasks are computationally feasible, i.e. polynomially tractable or at least decidable. As an example, propositional logic can be restricted to Horn clauses, which allow linear algorithms for satisfiability testing [28].

(2) According to [55], we could use a form of logic that allows weaker inferential power but is computationally feasible even with a full expressiveness in the language. As an example, propositional calculus without the *modus ponens* rule admits polynomial algorithms for consistency testing [36,54].

Both ideas received great attention both from the theoretical and from the application-oriented community. The limited expressiveness strategy underlies the whole database approach. The CLASSIC knowledge representation system [4,5], built at AT&T and currently used in an industrial environment, uses both strategies. CLASSIC adopts a restricted language—thus avoiding some sources of intractability—but allows constructors

that lead to intractable reasoning problems. Such constructors are dealt with incomplete inference algorithms grounded on non-standard semantics.

For what concerns the language restriction approach, fundamental studies on the complexity of fragments of classical propositional [66] and first-order logic [29] have been done in the last decades. More recently, computational studies about several logical formalisms relevant for KR appeared. Studies analyzing the so-called *tractability threshold* between polynomially tractable and intractable languages are of particular practical interest. Among the most significant in this group we cite [27] on concept description languages, [48,71] on default logic, [10] on closed-world reasoning and circumscription, [31] on logic-based abduction, [70] on path-based inheritance, [8] on set covering abduction.

As far as weak forms of reasoning are concerned, it is well known that AI has always dealt with approximation, incompleteness and heuristics. The kind of weak reasoning we are interested in in this work can be described in a nutshell as follows: we have a satisfactory logical formalization of a reasoning problem, but we don't want to implement it exactly as it is, because it is computationally too expensive; hence we look for a formalization that "looks like" our favorite one, but it is easier to compute.

In this work we are mainly interested in decision problems, as reasoning problems usually admit a boolean answer. Informally, an approximate solution to a decision problem is a "maybe" answer, equipped with reasons to believe that the "maybe" is actually a "yes" or to believe that it is actually a "no". In a form of approximate reasoning called *sound* reasoning we have two possible answers: "yes" and "maybe no". In the dual form of approximate reasoning (*complete* reasoning), the two possible answers are "no" and "maybe yes".

The obvious important questions we are faced with are: how do we measure the accuracy of an approximate answer? How do we know an approximate answer is any better than another one? It is important to recall that in logic we have no explicit metric that gives an immediate answer to the above questions. In this respect, approximation of reasoning problems is more difficult to study than approximation of optimization problems. Approximation schemata for reasoning problems are typically justified by means of cognitive or epistemic arguments (e.g. "this is an approximate but satisfactory description of how people reason").

Logic-based study of approximate reasoning is receiving increasing attention in the AI community. Several formalisms for weak reasoning that are supported by a "reasonable" semantics recently appeared. To this end the most significant approaches to a formal description of partial reasoning are Levesque's [54–56] architecture based on *incomplete* reasoning, Frisch's [35,36] *limited inference* systems, Crawford and Kuiper's [19] *access-limited* logic, Kautz and Selman's [47,49,69] *knowledge compilation* and Dean and Boddy's [23] *any-time* algorithms, further investigated by Russell and Zilberstein in [65] and by Ginsberg in [39].

In this work we are interested in the "tractability via approximation" approach. Our goal is to define a semantically well-founded logic for approximate reasoning, justifiable from the intuitive point of view, and to provide fast algorithms for dealing with it even when using expressive languages. We also want our logic to be useful to perform approximate reasoning in different contexts.

We define a method for the approximation of decision reasoning problems based on multivalued logics. The use of multivalued logics for representing forms of *local*, incomplete and polynomially tractable reasoning has already been attempted by other authors (cf. [36,54].) Our work expands and generalizes in several directions ideas presented by those researchers.

In order to introduce the topic of logic-based approximation, in this paper we survey two methods that have been defined in the literature by other authors: Levesque's *incomplete reasoning* and Selman and Kautz's *knowledge compilation*. The analysis of these methods motivates a list of desiderata for a new theory of approximation.

The major features of our technique are:

- approximate answers give semantically clear information about the problem at hand;
- approximate answers are easier to compute than answers to the original problem;
- approximate answers can be improved, and eventually they converge to the right answer;
- both sound approximations and complete ones are described.

Our method differs from the existing ones in one or more of the above points.

The method we propose is flexible enough to be applied to a wide range of reasoning problems. In our research we considered approximation of several decidable problems with different worst-case complexity, involving both propositional and first-order languages. In particular we defined approximation techniques for:

(1) propositional logic;
(2) fragments of first-order logic (concept description languages);
(3) modal logic.

In our research we also addressed the issue of representing the knowledge of a reasoner with limited resources and how to use such a knowledge for approximate reasoning purposes.

The structure of the paper is as follows: In Section 2 we survey two methods for approximate reasoning that have been defined in the literature by other authors. Our desiderata for a theory of approximation are given in Section 3. In Section 4 we formalize our technique and we illustrate it for entailment in propositional logic. In Section 5 we discuss the algorithmic aspects of such an approximation. In Section 6 we address the issue of representing in a modal language the approximate knowledge owned by a limited reasoner. In Sections 7 and 8 we apply the approximation technique to intractable tasks in fragments of first-order logic and propositional modal logics, respectively. In Section 9 we draw some conclusions, address open problems and sketch future research. All the proofs of the theorems are presented in appropriate appendices.

2. Logic-based methods for approximation: existing techniques

In this section we illustrate two logic-based methods for approximate reasoning and we make a brief comparison of them. In order to give the flavor of what is going on in the field, we show a "classic" method (Levesque's), which has been analyzed and used by several authors, and a new—but already popular—one (Selman and Kautz's).

2.1. Levesque's limited inference

The first method has been introduced by Levesque in several works [54–56] and it is based on the idea of *limited inference*. Independently, Frisch [35,36] developed similar formalizations.

Levesque argues that if we are to model the mental activity of an agent, it is not reasonable to assume that he always answers queries by using a method that needs an exponential amount of time. It is more realistic to assume that, when faced with normal problems, the agent only applies simple inference rules. On the other hand, when the agent is faced with extremely important or tricky questions, he moves to what Levesque calls a "puzzle mode" and uses more complex inference procedures. As a consequence, a knowledge representation and reasoning system should be able to distinguish between what is explicit or evident in what he knows, and what is implicit and can be inferred given enough time and motivation. Since ordinary logic does not make any distinction of this kind, it is necessary to make appropriate formal steps in this direction.

Levesque notices that a good deal of reasoning is based on detecting that a sentence and its negation are contradictory. As an example, starting from $(p \vee q) \wedge (\neg q \vee r)$, we infer that $(p \vee r)$ because q and $\neg q$ are contradictory. On the other hand we can think about a *shallow* reading of sentences, where the contradiction between a sentence and its negation is not observed. In particular, reasoning like in the above example is not possible in the shallow reading.

Levesque gives in [56] a formal definition—based on multivalued logics—of the notion of shallow reading of sentences. L denotes a set of propositional letters. A literal is a letter l of L or its negation $\neg l$. L^* denotes the set of all literals associated with the letters of L. A truth assignment is a function mapping the set of literals L^* into the set $\{0,1\}$.

Definition 2.1 (*Levesque* [56]). A 3-*interpretation* of L^* is a truth assignment which does not map both a letter l of L and its negation $\neg l$ into 0.

According to this definition, for each propositional letter l and each 3-interpretation I there are three possibilities (hence the name 3-interpretation):

(1) $I(l) = 1$ and $I(\neg l) = 0$;
(2) $I(l) = 0$ and $I(\neg l) = 1$;
(3) $I(l) = 1$ and $I(\neg l) = 1$.

Standard (2-valued) interpretations admit only the first two possibilities (where l is, respectively, *true* or *false*). The third possibility can be regarded as the logical value *contradiction*.

The notion of 3-interpretation is extended to formulae in Conjunctive Normal Form (CNF) in the following way: a 3-interpretation I *satisfies* a clause iff it maps one of its literals into 1; I satisfies a CNF formula iff it satisfies all of its clauses. In Section 4 we show how to define the 3-interpretation of an arbitrary formula.

The relation of 3-entailment can be immediately defined in the intuitive way:[2] T

[2] Our terminology and symbols are different from those used by Levesque.

3-*entails* γ (written $T \models^3 \gamma$) iff every 3-interpretation mapping T into 1 also maps γ into 1. Levesque notices that 3-entailment has interesting properties:

- *soundness*: for each T and γ, it holds that $(T \models^3 \gamma)$ implies $(T \models \gamma)$;
- *polynomiality*: if both T and γ are in CNF, then $T \models^3 \gamma$ can be tested in $O(|T| \cdot |\gamma|)$ time.

Another independent motivation for $\models^3$ is that $(T \models^3 \gamma)$ holds iff either γ is a tautology or T tautologically entails γ in the system of Relevance Logic of Anderson and Belnap and Dunn [2,30], as shown in [56]. Moreover $\models^3$ can be modeled in terms of the proof theory of propositional calculus without the *modus ponens* inference rule (see also [35,36]).

Example 2.2 (*Modus ponens does not hold for* 3-*entailment*). Let L be the set $\{a, b\}$, and T be $\{a, \neg a \vee b\}$. Clearly both $T \models^3 a$ and $T \models^3 \neg a \vee b$ hold, hence 3-entailment captures explicit knowledge. On the other hand $T \models b$ holds, but $T \models^3 b$ does not hold: the truth assignment which maps $a, \neg a$ and $\neg b$ into 1 and b into 0 is a 3-interpretation satisfying T.

Levesque's conclusion is that $\models^3$ can model the kind of quick surface reasoning that the agent should always do prior to any form of deep logical analysis or problem solving, i.e. before entering into the "puzzle mode".

2.2. *Selman and Kautz's Horn compilation*

The second method has been introduced by Selman and Kautz in [47,49,69] and it is based on the idea of *knowledge compilation*.

The starting point of the technique stems on the fact that testing $\Sigma \models \alpha$—where Σ and α are propositional formulae—is in general co-NP-complete, while it is doable in polynomial time when Σ is a Horn formula and α is in CNF. The fascinating question addressed by Selman and Kautz is the following: is it possible to *compile* a propositional formula Σ into a Horn one Σ' so that a significant amount of the inferences that are performed under Σ can be performed under Σ' in polynomial time?

Selman and Kautz notice that there exist two different ways of doing such a compilation. In the first case the compiled formula satisfies the relation $\Sigma' \models \Sigma$, or equivalently $\mathcal{M}(\Sigma') \subseteq \mathcal{M}(\Sigma)$—where $\mathcal{M}(\Phi)$ denotes the set of models of the formula Φ. For this reason Σ' is called a *Horn lower bound*—or LB—of Σ. As an example,[3] let Φ be the formula $(a \longrightarrow c) \wedge (b \longrightarrow c) \wedge (a \vee b)$. The formula $\Phi_{\text{lb}} = a \wedge b \wedge c$ is a Horn LB of Φ.

The second form of compilation is dual. The compiled version of Σ is a Horn formula Σ' that satisfies the relation $\Sigma \models \Sigma'$, or equivalently $\mathcal{M}(\Sigma) \subseteq \mathcal{M}(\Sigma')$. Σ' is called a *Horn upper bound*—or UB—of Σ. Returning to the previous example, the formula $\Phi_{\text{ub}} = (a \longrightarrow c) \wedge (b \longrightarrow c)$ is a Horn UB of Φ.

Compiled forms of a knowledge base can sometimes be used for providing a quick answer to an inference problem. As an example, if we are faced with the problem of checking $\Sigma \models \alpha$, we may benefit from the fact that for any Horn LB Σ_{lb} of Σ, $\Sigma_{\text{lb}} \not\models \alpha$

[3] Taken from [69].

implies $\Sigma \not\models \alpha$. Σ_{lb} is therefore a *complete approximation* of Σ. Dually, a Horn UB Σ_{ub} is a *sound approximation* of Σ, since $\Sigma_{\text{ub}} \models \alpha$ implies $\Sigma \models \alpha$.

Selman and Kautz define the notion of *Horn greatest lower bound* or GLB of a formula Σ, which is a Horn formula Σ_{glb} such that $\mathcal{M}(\Sigma_{\text{glb}}) \subseteq \mathcal{M}(\Sigma)$ and for no Horn formula Σ' it holds that $\mathcal{M}(\Sigma_{\text{glb}}) \subset \mathcal{M}(\Sigma') \subseteq \mathcal{M}(\Sigma)$. A Horn GLB is in some sense the "best" complete Horn approximation. Continuing the previous example, $\Phi_{\text{lb2}} = a \wedge c$ is a Horn GLB of Φ.

Dually, a *Horn least upper bound* or LUB of a formula Σ is a Horn formula Σ_{lub} such that $\mathcal{M}(\Sigma) \subseteq \mathcal{M}(\Sigma_{\text{lub}})$ and for no Horn formula Σ' it holds that $\mathcal{M}(\Sigma) \subseteq \mathcal{M}(\Sigma') \subset \mathcal{M}(\Sigma_{\text{lub}})$. $\Phi_{\text{ub2}} = c$ is a Horn LUB of Φ.

Selman and Kautz's proposal is to approximate inference under a propositional formula Σ by using its Horn GLBs and LUBs. In this way inference could be unsound or incomplete, but it is anyway possible to spend more time and use a general inference procedure to determine the answer directly from the original theory. The general inference procedure could still use the approximations to prune its search space.

We now summarize the major properties of Horn GLBs and LUBs stated in [47,49, 69]:

- *size* of the formula with respect to the size $|\Sigma|$ of Σ:
  - GLB: linear;
  - LUB: in general exponential;
- *number* of possible approximations of this kind:
  - GLB: one;
  - LUB: many;
- *computational complexity* of the search problem of finding the approximation:
  - GLB: NP-hard;
  - LUB: NP-hard.

Other authors [9,41] have further explored the Horn compilation method, analyzing several computational properties. Relations between Horn compilation and nonmonotonic reasoning are shown in [9].

2.3. Comparison between the two approaches

Levesque's approach is characterized by philosophical motivations and aims at building inference systems justifiable from the intuitive point of view. Semantics is important for Levesque, as well as a comparison to techniques for the formalization of limited reasoning developed in different fields (e.g. Relevance Logic [2]). Selman and Kautz's method is motivated with strictly computational arguments, and semantics of approximate answers is given in terms of the syntactic notion of Horn clause.

Knowledge compilation supports sound inferences as well as complete ones. Levesque's method is only sound.

Levesque's inference is done in polynomial time. Both kinds of Horn compilation are polynomially intractable and the Horn LUB needs exponential space.

While knowledge compilation can be done off-line, Levesque's incomplete inference has to be performed on-line. Compiled knowledge bases can be used for answering many queries.

Both methods are characterized by being "one-shot": if we know that the approximate solution may be wrong, the sole thing to do is to resort on general-purpose theorem-proving procedures. In a nutshell: the quality of approximate solutions is not improvable.

Levesque's idea has been generalized to the solution of other reasoning problems, like epistemic reasoning [53, 54] and reasoning in terminological languages [61].

3. Guidelines of a new method for approximation

Here is our desiderata list for a theory of approximate reasoning. We would like a method that is:

- *semantically well-founded*: approximate answers should give semantically clear information about the problem at hand;
- *computationally attractive*: approximate answers should be easier to compute than answers to the original problem;
- *improvable*: approximate answers can be improved, and eventually they converge to the right answer (provided we have enough time and motivation);
- *dual*: both sound approximations and complete ones should be described;
- *flexible*: the approximation schema should be general enough to be applied to a wide range of reasoning problems.

We found in Levesque's approach inspiration for developing a method that fulfills all the above desiderata. Loosely speaking we added to Levesque's method duality and improvability, keeping its nice computational features and its semantical flavor coming from a clear formulation in multivalued logics. We give now an abstract description of our technique.

We model a reasoning task as the problem of deciding whether a string x belongs to a set $\mathcal{D}$ (e.g. the set of satisfiable propositional formulae) or not. The method we defined for approximating a reasoning problem, or equivalently the corresponding set $\mathcal{D}$, relies on the definition of two sequences of sets $\langle \mathcal{D}_0, \mathcal{D}_1, \ldots, \mathcal{D}_m \rangle$ and $\langle \mathcal{D}^0, \mathcal{D}^1, \ldots, \mathcal{D}^n \rangle$ such that

$$\mathcal{D}_0 \subseteq \mathcal{D}_1 \subseteq \cdots \subseteq \mathcal{D}_m = \mathcal{D} = \mathcal{D}^n \subseteq \mathcal{D}^{n-1} \subseteq \cdots \subseteq \mathcal{D}^0. \tag{1}$$

The sequences have the following properties:

- The lengths n and m of the sequences are polynomial with respect to the input x of the problem.
- The elements of both sequences are defined by means of a semantics closely related to that of $\mathcal{D}$.
- Deciding whether the input string x belongs to $\mathcal{D}^0$ or not (or loosely, deciding membership in $\mathcal{D}^0$) is a polynomial problem; the same holds for $\mathcal{D}_0$. In general deciding membership in $\mathcal{D}^i$ and in $\mathcal{D}_i$ gets exponentially harder as i grows, but it is not harder than deciding membership in $\mathcal{D}$.

The reasoning task is performed in an incremental fashion, by deciding membership in sets $\mathcal{D}_i$ and $\mathcal{D}^j$ of both sequences for increasing indexes i and j, starting with $i = j = 0$. If we prove membership in any $\mathcal{D}_i$, then we have also proved membership in $\mathcal{D}$; on

the other hand if we disprove membership in any $\mathcal{D}^j$, then we have also disproved membership in $\mathcal{D}$.

There are clearly two possibilities when we use this method: either we are able to solve the reasoning problem in a reasonable amount of time (in general for small indices i and j), or i and j get too large and we run out of computing resources. Since the reasoning problems taken into account are polynomially intractable, the latter case will be frequent, therefore it has to be analyzed very carefully.

The whole purpose of our research on approximation is the study of how to give a meaningful answer in the latter case. We believe that such a meaningful answer has to be grounded on a simple semantics and justified by intuitive arguments. In this way we want to obtain a comprehensible reasoning system whose precision is arbitrary, but depends on the computational effort that has been spent. The system is realized by means of a stepwise procedure that can be interrupted at any step in such a way that the information obtained so far gives interesting semantic insights.

In all applications of the above approximation schema presented throughout the paper, the lengths n and m of the sequences are linear with respect to the input of the problem.

Patel-Schneider [61,62] and Frisch [36] pointed out that Levesque's system successfully managed to lower the complexity of the inference tasks, but it is far too weak in sanctioning conclusions to be useful in many situations (see Example 2.2). They also argued that Levesque's weak semantics is the building block on top of which more inferential capabilities should be added, without losing tractability. This is exactly the direction of research we pursue in our work. What we will show is that:

- a very weak semantics can be used as the starting step of high complexity decision problems;
- on top of this we can define sound and increasingly more complete procedures that are based on stronger semantics.

In the following we also present a dual semantics being stronger than the classical one, that will be used as a basis for complete and increasingly more sound reasoning procedures. What we gain with our approach is the ability to characterize with a precise semantics a wide class of incomplete or unsound inference procedures.

We proved the flexibility of the method applying it to several reasoning problems very popular in knowledge representation. In particular we defined approximation methods for:

(1) propositional logic;
(2) fragments of first-order logic (concept description languages);
(3) propositional default logic and circumscription;
(4) modal logics.

Moreover we defined an epistemic language for the representation of resource-bounded reasoners. The language provides for a meta-level description of the approximate knowledge owned by an agent. In the rest of this paper we show all such applications, except for approximations of propositional nonmonotonic logics, that we have already analyzed in [14].

Our method depends on the complexity of the reasoning task only to some extent. The complexity of the reasoning problems addressed ranges from NP-complete to PSPACE-complete.

4. Propositional logic

In this section we illustrate our approximation technique for entailment in propositional logic. In particular we define a method for approximating the set

$$\{\langle T;\gamma\rangle \mid T \text{ and } \gamma \text{ are propositional formulae and } T \models \gamma\}.$$

In Section 4.1 we define the semantics of the approximation. In Section 4.2 we address computational aspects and in Section 4.3 we present some examples and discuss the method.

4.1. Semantics

As we said in the previous section, we are interested both in sound and in complete approximations. We first show the sound approximation, which is a generalization of Levesque's shallow reading of sentences (cf. Definition 2.1). Throughout this section we assume that there is an underlying finite alphabet L used for building all the sentences. Symbols t and f are used for denoting special propositional letters, which are always mapped into 1 and 0, respectively. In the following we denote with S a subset—possibly not proper—of L.

Definition 4.1 (*S-3-interpretation*). An S-3-interpretation of L is a truth assignment which maps every letter l of S and its negation $\neg l$ into opposite values. Moreover, it does not map both a letter l of $L \setminus S$ and its negation $\neg l$ into 0.

According to this definition, for each propositional letter $l \in L$ and each S-3-interpretation I there are the following possibilities:

(1) $I(l) = 1$ and $I(\neg l) = 0$;
(2) $I(l) = 0$ and $I(\neg l) = 1$;
(3) $I(l) = 1$ and $I(\neg l) = 1$ [only if $l \in L \setminus S$].

S-3-interpretations are therefore "something in between" standard 2-valued interpretations and Levesque's 3-interpretations. Often we refer to standard 2-valued interpretations using the term *2-interpretations*. Both 2-interpretations and 3-interpretations are generalized by S-3-interpretations, since a 2-interpretation is an S-3-interpretation with $S = L$, while a 3-interpretation is an S-3-interpretation with $S = \emptyset$. We can say that in S-3-interpretations the truth value *contradiction* is possible only for those letters that do not belong to S.

Interpretations (standard 2-valued interpretations as well as 3-interpretations or S-3-interpretation) assign a truth value to an arbitrary formula by means of simple rules. First of all, we put formulae in Negation Normal Form (NNF) using the following rewriting rules:

$$\neg(\alpha \wedge \beta) \mapsto \neg\alpha \vee \neg\beta,$$
$$\neg(\alpha \vee \beta) \mapsto \neg\alpha \wedge \neg\beta,$$
$$\neg\neg(\alpha) \mapsto \alpha,$$
$$(\alpha \rightarrow \beta) \mapsto (\neg\alpha \vee \beta).$$

Now negation occurs in a formula only at the literal level. Rules for assigning truth values to NNF formulae are the following:

- $\vee$-*rule*: $I \models \alpha \vee \beta$ iff $I \models \alpha$ or $I \models \beta$;
- $\wedge$-*rule*: $I \models \alpha \wedge \beta$ iff $I \models \alpha$ and $I \models \beta$.

A formula is S-3-*satisfiable* if an S-3-interpretation I exists such that I satisfies it, i.e. I maps it into 1. Entailment is defined in the standard way for S-3-entailment: we say that T S-3-*entails* γ (written $T \models^3_S \gamma$) iff every S-3-interpretation satisfying T also satisfies γ.

Example 2.2 of Section 2.1 showed that the *modus ponens* rule does not hold for 3-entailment. Let us continue that example considering the definitions we just gave.

Example 4.2 (*Sometimes modus ponens holds for S-3-entailment* ...). Let L be the set $\{a, b\}$, and T be $\{a, \neg a \vee b\}$. $T \models b$ holds and $T \models^3 b$ does not hold. If $S = \{a\}$, then $T \models^3_S b$, since every S-3-interpretation satisfying T maps a and b into 1, and $\neg a$ into 0.

(... *and sometimes it doesn't*). Let L be the set $\{a, b, c\}$, $S = \{a\}$ and T be $\{a, \neg a \vee b, \neg b \vee c\}$. $T \models c$ holds, but $T \models^3_S c$ does not hold.

In the above example the relation $\models^3_S$ is *sound* and not complete with respect to $\models$. We will see in the following that this is a general property of $\models^3_S$ and we will return later on to the possibility of using $\models^3_S$ as an approximation of $\models$. Now we want to deal with *complete and unsound* approximations of $\models$. Such a form of entailment is based on the following definition.

Definition 4.3 (S-1-*interpretation*). An S-1-interpretation of L is a truth assignment which maps every letter l of S and its negation $\neg l$ into opposite values. Moreover, it maps every letter l of $L \setminus S$ and its negation $\neg l$ into 0.

According to this definition, for each propositional letter $l \in L$ and each S-1-interpretation I there are the following possibilities:

(1) $I(l) = 1$ and $I(\neg l) = 0$;
(2) $I(l) = 0$ and $I(\neg l) = 1$;
(3) $I(l) = 0$ and $I(\neg l) = 0$ [if and only if $l \in L \setminus S$].

The reason why every letter l of $L \setminus S$ and its negation $\neg l$ are mapped into 0 will be clarified shortly. The name S-1-interpretation originates from the fact that for each letter $l \in L \setminus S$ there is exactly one possibility. Intuitively, letters in $l \in L \setminus S$ are mapped into a truth value which is different from those already seen (*true, false, contradiction*), that we might call *undefined*. Truth assignment for arbitrary formulae is defined in the same way we did for S-3-interpretations. Analogously, S-1-entailment (denoted by $\models^1_S$) directly follows from that of S-1-interpretation using the same rule. We note that 2-interpretations are not S-1-interpretations and vice versa, unless $S = L$. In the following we show an example in which the notions of S-1-interpretation and S-1-entailment are used.

4.2. Computational aspects

In this subsection we show some properties of S-3- and S-1-entailment that are important from the computational point of view. We denote with T a generic propositional

CNF formula and with γ a generic propositional clause not containing both a letter l and its negation $\neg l$.

Theorem 4.4 (Monotonicity). *For any S and S' such that $S \subseteq S' \subseteq L$, if $T \not\models^1_S \gamma$, then $T \not\models^1_{S'} \gamma$ (hence $T \not\models \gamma$). Moreover if $T \models^3_S \gamma$, then $T \models^3_{S'} \gamma$ (hence $T \models \gamma$).*

Observation 4.5 (Convergence). *If $T \models \gamma$, then there exists an $S \subseteq L$ such that $T \models^3_S \gamma$. If $T \not\models \gamma$ then there exists an $S \subseteq L$ such that $T \not\models^1_S \gamma$.*

Theorem 4.6 (Uniform complexity). *There exists an algorithm for deciding if $T \models^3_S \gamma$ and deciding if $T \models^1_S \gamma$ which runs in $O(|T| \cdot |\gamma| \cdot 2^{|S|})$ time.*

We notice that Observation 4.5 directly follows from the previous definitions. A set S that trivially satisfies its statement is L itself, but we are interested in sets S of smaller size. Theorem 4.4 and Observation 4.5 show that the co-NP-complete problem of deciding whether $T \models \gamma$ holds can be computed in a stepwise fashion by proving or disproving $T \models^1_S \gamma$ and $T \models^3_S \gamma$ for increasing sets S, starting with $S = \emptyset$ and stopping for the least $S \subseteq L$ such that either $T \not\models^1_S \gamma$ or $T \models^3_S \gamma$ hold.

Referring to the terminology introduced in Section 3 (cf. formula (1)), we are approximating the set

$$\mathcal{D} = \{\langle T; \gamma\rangle \mid T \models \gamma\}$$

by means of the two families of sets

$$\mathcal{D}_i = \{\langle T; \gamma\rangle \mid T \models^3_{S_i} \gamma\}, \qquad \mathcal{D}^j = \{\langle T; \gamma\rangle \mid T \models^1_{S^j} \gamma\}.$$

In particular an approximation is defined by two increasing sequences of sets

$$\langle S_0 = \emptyset \subset \cdots \subset S_i \subset \cdots \subset S_m = L\rangle,$$

$$\langle S^0 = \emptyset \subset \cdots \subset S^j \subset \cdots \subset S^n = L\rangle.$$

The above results do not tell us for which set S we are going to have a definite answer to the query $T \models \gamma$. Theorem 4.6 tells us that, even in the worst case (i.e. when we prove or disprove $T \models \gamma$ for $S = L$), the complexity of the method is $O(|T| \cdot |\gamma| \cdot 2^{|L|})$, hence similar to the best-known algorithms for deciding $T \models \gamma$.

Notice that if we change Definition 4.3 so that an S-1-interpretation I satisfies the weaker requirement

(3′) $I(l) = 0$ and $I(\neg l) = 0$ [only if $l \in L \setminus S$],

then deciding whether $T \models^1_S \gamma$ would be a co-NP-complete problem for all sets S. As a matter of fact, $T \models^1_S f$ iff T is not S-1-satisfiable, and T would be S-1-satisfiable, according to the revised definition, iff T is classically (2-valued) satisfiable.

The method for deciding propositional entailment can be stopped for any $S \subset L$, and in this case the time spent in the computation has provided clear semantical information. Propositional entailment is computed in a step-wise fashion, and the parameter S controls

the quality of the inference. The precision of the inference is arbitrary and depends on the computational effort that has been spent. If $|S|$ is limited by a logarithmic function, then the resulting inference is polynomial. This is a typical *approximation* process, since every intermediate step provides a partial solution whose relation to the final solution (the *error*) is clearly identified.

We have therefore reached our goal of defining a method for approximation of the consequence relation in propositional logic such that:

- it is semantically founded;
- approximate answers are easier to compute than the answers to the original inference problem;
- approximate answers can be improved, and converge to the right answer (provided we have enough resources and motivation);
- both only sound and only complete approximations are provided.

In the rest of the paper we show methods for approximating other reasoning tasks. All such methods are supported by results analogous to Theorems 4.4 and 4.6 and Observation 4.5.

Results analogous to Theorems 4.4 and 4.6 and Observation 4.5 hold for S-1- and S-3-satisfiability problems. This will be the topic of Section 5.

All the results just shown (and in particular Theorem 4.6) hold even if T is a NNF formula and γ is a generic formula in CNF. This aspect has been analyzed in [15], where we showed other normal forms for which the uniform complexity result holds.

4.3. Discussion and examples

In this subsection we perform some informal considerations on the meaning of the entailment relations we defined. Let's start considering two examples.

Example 4.7 (*Proving a consequence*). We assume that the following CNF formula T is part of a very large knowledge base containing information about animals and their properties. T uses a small alphabet L, which we assume to be part of a larger dictionary. The large knowledge base contains a taxonomy of classes, which is partially present in T.

$$L = \{\, cow, dog, grass\text{-}eater, carnivore, mammal, has\text{-}canine\text{-}teeth, has\text{-}molar\text{-}teeth, vertebrate, animal \,\};$$

$$\begin{aligned} T = {} & (\neg cow \vee grass\text{-}eater), \\ & (\neg dog \vee carnivore), \\ & (\neg grass\text{-}eater \vee \neg has\text{-}canine\text{-}teeth), \\ & (\neg grass\text{-}eater \vee mammal), \\ & (\neg carnivore \vee mammal), \\ & (\neg mammal \vee has\text{-}canine\text{-}teeth \vee has\text{-}molar\text{-}teeth), \\ & (\neg mammal \vee vertebrate), \\ & (\neg vertebrate \vee animal). \end{aligned}$$

Notice that T is not a Horn formula. We want to prove that $T \models (\neg cow \vee has\text{-}molar\text{-}teeth)$ holds. Since $T \not\models^3 (\neg cow \vee has\text{-}molar\text{-}teeth)$, we try to determine a subset S of L such that $T \models^3_S (\neg cow \vee has\text{-}molar\text{-}teeth)$ holds. By Theorem 4.4 this is sufficient for our goal. In fact, this happens when $S = \{grass\text{-}eater, mammal, has\text{-}canine\text{-}teeth\}$, which is a small subset of the whole dictionary.

Example 4.8 (*Disproving a consequence*).

$$L = \{ person, child, youngster, adult, senior, student, pensioner, worker, unemployed \};$$

$$\begin{aligned} T = {} & (\neg person \vee child \vee youngster \vee adult \vee senior), \\ & (\neg youngster \vee student \vee worker), \\ & (\neg adult \vee student \vee worker \vee unemployed), \\ & (\neg senior \vee pensioner \vee worker), \\ & (\neg student \vee child \vee youngster \vee adult), \\ & (\neg pensioner \vee senior), \\ & (\neg pensioner \vee \neg student), \\ & (\neg pensioner \vee \neg worker). \end{aligned}$$

We want to prove that $T \not\models (\neg child \vee pensioner)$. By Theorem 4.4 we try to determine a subset S of L such that $T \not\models^1_S (\neg child \vee pensioner)$. In fact this happens when $S = \{child, worker, pensioner\}$ since the S-1-interpretation mapping *child*, *worker*, $\neg pensioner$ into 1 and all the other literals into 0 satisfies T but not $(\neg child \vee pensioner)$.

Let us give a qualitative analysis of the above examples. In the first example, in which taxonomic knowledge is present, it seems that in order to reach an exact solution it is useful to include in the set S concepts of the taxonomy which are superclasses of the concepts occurring in the query. As for the second example, in which the knowledge is spread over several indefinite clauses, an useful strategy seems to be that of extending S with classes with several properties, such as *pensioner*, or properties shared by several classes, such as *worker*.

From the intuitive point of view, both S-3-interpretation and S-1-interpretation correspond to a representation in which only some of the propositional letters are "taken seriously", while others are ignored. This leads to an entailment relation that models a way of reasoning in which part of the knowledge (the knowledge represented by letters not in the set S) is ignored on purpose. S-3- and S-1-entailment are therefore mechanisms for creating partial *views* of the knowledge base.

S-3-entailment and S-1-entailment differ with respect to the treatment of the letters that are ignored. In particular in S-3-entailment we don't care if ignoring letters leads to contradictions, i.e. a knowledge base T may have plausible states (S-3-models) that do not correspond to ordinary models. This proliferation of admissible states makes it "harder" (logically, not computationally) to prove entailment, i.e. S-3-entailment is sound and not complete with respect to classical entailment.

We claim that S-3-entailment models a deductive strategy that is sometimes applied in reality. Let us see this with an example, paraphrasing Example 4.7. We are consulting an encyclopedia about animals, looking for particular information, let's say whether cows have molar teeth or not. It is natural to start our research from the paragraph about cows, then to broaden the scope more and more, taking into account the section on herbivores, the chapter on mammals, the volume on vertebrates, ... During this process it may happen that we find a term that we do not understand, because it is defined in a part that we did not read. Anyway we continue until we are able to answer the question or we run out of resources (time, motivation, ...). We are performing two different forms of local reasoning: (1) only a part of the knowledge base is taken into account and (2) only a part of the alphabet is taken into account. While performing this research, we assume that the information not yet considered can be ignored, in the sense that if we prove that cows actually have molar teeth by considering only little information or a small part of the alphabet, then such an answer is definitely exact. Ignoring a part of the knowledge base (i.e. the first aspect of local reasoning) can be trivially modeled. On the other hand S-3-entailment models the second aspect of local reasoning: the set S represents the terms that "make sense" to us. Broadening the scope of our research is modeled by making the set S larger and larger.

As far as the S-1-entailment relation is concerned, dual considerations hold. Also in this case we deal with a form of local reasoning, but the information which is ignored is considered in a pessimistic way. In other words we think that the information we are currently holding is useful only for disproving, and not for proving. This leads to dramatic reduction of the admissible states of a knowledge base (i.e. its S-1-models), hence it is possible to accept as a consequence what is not really a consequence (S-1-entailment is complete and unsound with respect to 2-entailment).

Summing up, both relations for approximate entailment can model agents with limited resources. The difference is in that they have different attitudes with respect to what is outside the scope of the agents' competence.

Examples 4.7 and 4.8 show that an exact solution to an entailment problem can be reached with a small S. The choice of the minimal set S having this property may not be easy in general. If we knew the minimum size of a set S for which $T \models^3_S \gamma$ or $T \not\models^1_S \gamma$ holds, then we would know that an upper bound for the entailment problem is $O(|T| \cdot |\gamma| \cdot 2^{|S|})$.

A natural question is therefore the following: is approximation of propositional entailment only a theoretical method, or do we have a technique for choosing a set S of letters for which we have high expectation of having correct answers? Moreover, how much can we trust an approximate answer? What kind of "degree of belief" can we associate to it?

There are several ways to give answers to this kind of questions. For example we could look for "good" sets S using purely syntactic techniques, like using topological criteria in suitable representations of the formula as a graph. Another possibility relying on syntactic methods is to use parameters such as the number of occurrences of a single literal in the formula. Techniques of this kind have actually been used by several researchers for providing smart methods for solving computationally hard problems (see for example [24] in the area of constraint propagation). Moreover several heuristics for

an intelligent choice of the letters to be used in resolution algorithms (see for example [57]) or in Davis–Putnam's procedure for propositional satisfiability (see for example [46]) have been developed. Another possibility is to use numerical parameters for the attribution of a degree of confidence to an approximate answer.

In our work we preferred to use a logical language for addressing the questions reported above. We deal with these aspects in Section 6, where preliminary answers have been given in the framework of epistemic logic. Our goal has been to provide a meta-level language for the representation of approximate knowledge, as specified by the logical relations presented in this section. This meta-level knowledge can be used for a smart choice of the set S.

Techniques defined by other authors—apart from those addressed in Section 2—share ideas with our method. For example Giunchiglia and Walsh's *abstract proofs* method [40] give logical description of only sound as well as only complete forms of approximate reasoning. Related ideas can also be found in Imielinski's work on *domain abstraction* [44] and in general in the area of *abstract interpretation*, born in the programming languages community [18]. Gallo and Scutellà's method [37], modified by Dalal and Etherington in [21], defines classes of propositional formulae whose associated satisfiability problem is more and more computationally demanding. Dalal's *k-consistency* [20] is similar to this respect.

Our method differs significantly from all the above techniques.

All the results presented in this section will be proved in Appendix A.1.

The ideas shown in this section have been presented in a preliminary form in [11].

5. Algorithms to compute S-entailment

The goal of this section is to discuss algorithmic aspects of S-3- and S-1-entailment. Taking into account that $T \models \gamma$ holds iff $T \cup \{\neg\gamma\}$ is unsatisfiable—i.e. for 2-valued semantics entailment can be reduced to unsatisfiability—in the first subsection we develop methods for testing S-1- and S-3-entailment based on S-1- and S-3-unsatisfiability, respectively. In the second subsection we focus on the development of an algorithm for checking S-1- and S-3-satisfiability of a formula.

5.1. Reducing S-entailment to S-unsatisfiability

In this subsection we use the symbol γ to denote a clause. First of all, we can immediately reduce S-unsatisfiability to S-entailment, since a formula T is S-1-satisfiable if and only if $T \not\models^1_S f$, and T is S-3-satisfiable if and only if $T \not\models^3_S f$.

As for the reverse reduction, we introduce a set describing the clause γ.

Definition 5.1. We denote with $letters(\gamma)$ the set $\{l \in L \mid l \text{ occurs in } \gamma\} \cup \{l \in L \mid \neg l \text{ occurs in } \gamma\}$.

The next lemma shows that, when dealing with S-3-entailment, we can safely choose an S such that $letters(\gamma) \subseteq S$.

Lemma 5.2. *Suppose* $letters(\gamma) \not\subseteq S$ *holds. Let* S' *be the set* $S \cup letters(\gamma)$. $T \models^3_S \gamma$ *holds iff* $T \models^3_{S'} \gamma$ *holds.*

The next two theorems show that S-1- and S-3-entailment can be reduced to S-1- and S-3-unsatisfiability, respectively.

Theorem 5.3 (Reducing S-1-entailment to S-1-unsatisfiability). *Let* γ *be* $\gamma_S \vee \gamma_{\overline{S}}$, *where both* $letters(\gamma_S) \subseteq S$ *and* $letters(\gamma_{\overline{S}}) \cap S = \emptyset$ *hold.* $T \models^1_S \gamma$ *holds iff* $T \cup \{\neg\gamma_S\}$ *is not* S-1-*satisfiable.*

Theorem 5.4 (Reducing S-3-entailment to S-3-unsatisfiability). *Let* $letters(\gamma) \subseteq S$ *hold.* $T \models^3_S \gamma$ *holds iff* $T \cup \{\neg\gamma\}$ *is not* S-3-*satisfiable.*

The above theorems show that the problem of testing S-3-entailment [S-1-entailment] can be reduced to the problem of deciding S-3-unsatisfiability [S-1-unsatisfiability] of a suitable formula. This extends the well-known relation existing between classical entailment and unsatisfiability. The importance of such a result is in that S-3-satisfiability [S-1-satisfiability] of a CNF formula T can be tested in the following way:

(1) replace by t [f] all occurrences (both positive and negative) in T of letters which belong to $L \setminus S$, thus obtaining the formula T^3_S [T^1_S];
(2) test standard (2-valued) satisfiability of T^3_S [T^1_S].

The transformation shown in point (1) preserves S-3- and S-1-satisfiability for NNF formulae as well (see [15]). We notice that only letters which belong to S occur in T^3_S and T^1_S. As a consequence simple algorithms for deciding S-1- and S-3-satisfiability and running in time $O(|T| \cdot 2^{|S|})$ can be obtained directly from classic algorithms for satisfiability like Davis and Putnam's [22] or Robinson's [63]. Nevertheless very specialized algorithms for propositional satisfiability (e.g. Van Gelder's [72]) can be used without loss of efficiency for computing S-1- or S-3-satisfiability.

We want to stress that we are not proposing a new algorithm or heuristic for the satisfiability problem, but we are instead interested in a semantic description of *partial* reasoning. We claim that many heuristics commonly used for the satisfiability problem can be applied in our method as well, since we are able to reduce the relations for approximate entailment to ordinary (2-valued) satisfiability of a smaller formula.

In the next subsection we show custom algorithms for deciding S-1- and S-3-satisfiability.

We now present some simple observations leading to significant simplifications of the formulae whose S-1- and S-3-unsatisfiability we must check in order to prove S-1- and S-3-entailment.

We define an operation on formulae; we say that $\Theta = simplify(\Gamma, \delta)$, where Θ and Γ are CNF formulae and δ is a set or a conjunction of literals, if Θ is obtained from Γ through the following steps:

(1) delete all clauses of Γ that contain a literal occurring in δ;
(2) in any clause β of Γ delete all literals l whose negation $\neg l$ occurs in δ.

This operation can be done in $O(|\Gamma| \cdot |\delta|)$ time. Notice that no literal p of δ or its negation $\neg p$ occur in Θ. Moreover, the simplified CNF formula Θ is always smaller than the original formula Γ, since some of the clauses of the latter disappeared, and some of the remaining clauses contain less literals. We can now prove a simple result about simplified formulae, stating that the simplifying process does not lead to loss of information.

Theorem 5.5 (Simplifying). *Let δ be a set of literals such that letters(δ) $\subseteq S$ holds and both a letter l and its negation $\neg l$ do not occur in δ; simplify(T, δ) is S-1-satisfiable iff $T \cup \delta$ is S-1-satisfiable. In addition, simplify(T, δ) is S-3-satisfiable iff $T \cup \delta$ is S-3-satisfiable.*

The above theorem is intuitive when $S = L$ holds, that is, for the 2-valued logic. By taking into account Theorems 5.3 and 5.4, the above result shows that, in order to prove whether $T \not\models^1_S \gamma$ and $T \models^3_S \gamma$ holds, we can focus our attention on S-1-satisfiability of *simplify*($T, \neg\gamma_S$) and on S-3-unsatisfiability of *simplify*($T, \neg\gamma$), respectively. The role of simplification of T is that of reducing its dimensions, thus gaining efficiency in the whole process of determining the validity of the entailment relation $\models$. Anyway, the simplification can be done if the following hypotheses hold:

(1) γ is not a tautology;
(2) *letters*(γ) $\subseteq S$ [only for $\models^3_S$].

Since checking entailment of tautologies is a trivial problem, the first hypothesis can be assumed without loss of generality. Lemma 5.2 shows that also the second hypothesis can be assumed without loss of generality.

5.2. Algorithms for testing S-satisfiability

As we showed in the previous subsection, testing S-entailment can always be reduced to testing S-satisfiability. In this subsection we focus on the development of an algorithm for checking S-1- and S-3-satisfiability. In the first part we show the relations existing between S-satisfiability and well-known algorithms for satisfiability. In the second part we design an original algorithm.

5.2.1. Resolution and enumeration

Satisfiability of CNF formulae is a very well-known problem in 2-valued propositional logic, extensively studied and discussed in the specialized literature. Here we focus on two of the major types of algorithms for satisfiability:

(1) *Resolution-based algorithms* (e.g. [63]): they try to prove unsatisfiability of a formula T by deriving the empty clause from it. If the empty clause cannot be derived, then T is satisfiable.
(2) *Enumeration-based algorithms* (e.g. [22]): they try to prove satisfiability of a formula T by generating an interpretation which satisfies T. If such an interpretation is not found, then T is unsatisfiable.

The first type of algorithms is more directed to proving unsatisfiability, hence we expect them to be well suited to check S-3-unsatisfiability, while we expect the latter ones to be well suited to check S-1-satisfiability. We now state formally the correspondence

between these algorithms and the checks we are interested in. In the following we denote with $\square$ the empty clause, and with Ω the CNF formula with no clauses.

In order to show the link existing between S-satisfiability and resolution-based algorithms, we introduce a family of CNF formulae T^i $(1 \leqslant i \leqslant m)$, defined as follows (we denote with $\{a_1, \ldots, a_m\}$ the set S):

$$T^0 = T;$$

$$T^{i+1} = \{x \mid \exists y_1, y_2 \in T^i \text{ and } x \text{ is the resolvent of } y_1 \text{ and } y_2 \text{ upon } a_{i+1}\} \cup \{x \mid x \in T^i,\ a_{i+1} \notin x \text{ and } \neg a_{i+1} \notin x\}.$$

The CNF formula T^i $(1 \leqslant i \leqslant m)$ is the conjunction of all the resolvents at the ith level of the resolution tree having T as the root and which have been produced by resolving upon all the literals in $\{a_1, \ldots, a_i\}$ in any order. Notice that T^i has $O(2^i)$ times the number of clauses T has. Notice also that no literal of $\{a_1, \ldots, a_i\}$ occurs in T^i. The next theorem shows how S-satisfiability of a CNF formula T can be computed using the resolution method.

Theorem 5.6 (S-satisfiability and resolution). *A formula T is S-3-satisfiable iff T^m is S-3-satisfiable iff $\square \notin T^m$. A formula T is S-1-satisfiable iff T^m is S-1-satisfiable iff $T^m = \Omega$.*

Similar considerations enable us to relate S-satisfiability of a CNF formula T to the Davis-Putnam algorithm for checking satisfiability [22]. For this aim we introduce a family Φ^i $(1 \leqslant i \leqslant m)$ of sets of CNF formulae defined as follows:

$$\Phi^0 = \{T\};$$

$$\Phi^{i+1} = \{W \mid \exists H \in \Phi^i \text{ and } W = \mathit{simplify}(H, \{a_{i+1}\})\} \cup \{W \mid \exists H \in \Phi^i \text{ and } W = \mathit{simplify}(H, \{\neg a_{i+1}\})\}.$$

The family Φ^i $(1 \leqslant i \leqslant m)$ is the set of all the CNF formulae H obtained by simplifying T with any set of literals representing a 2-interpretation of the letters in $\{a_1, \ldots, a_i\}$. In the generation of Φ^{i+1} any formula H belonging to Φ^i is replaced by the two formulae $\mathit{simplify}(H, \{a_{i+1}\})$ and $\mathit{simplify}(H, \{\neg a_{i+1}\})$. Notice that Φ^i has 2^i CNF formulae, each being smaller than T.

Theorem 5.7 (S-satisfiability and Davis–Putnam algorithm). *A formula T is S-3-satisfiable iff there exists a $H \in \Phi^m$ that is S-3-satisfiable; H is S-3-satisfiable iff $\square \notin H$. A formula T is S-1-satisfiable iff there exists a $H \in \Phi^m$ that is S-1-satisfiable; H is S-1-satisfiable iff $H = \Omega$.*

The possibility of using Theorems 5.6 and 5.7 for making an appropriate choice of the set S will be discussed shortly.

5.2.2. The algorithm

We briefly sketch our desiderata about an algorithm for testing S-satisfiability:

- The algorithm must compute at the same time both S-1-satisfiability and S-3-unsatisfiability of a generic formula.
- It should benefit from previous computations. In other words, if for a given S, T is still S-1-unsatisfiable and S-3-satisfiable, we probably want to try with a set $S' \supset S$. In principle this may happen after hours or days, depending on our computing resources; therefore we want our algorithm to compute satisfiability *incrementally*, using information gained in previous steps.
- Although we expect the algorithm to run in time exponential in $|S|$, we want the algorithm to use polynomial space.

By taking into account that we want to accommodate a stepwise extension of the set S, an important issue for an efficient algorithm is that this extension does not require to perform all the computations from scratch. In other words, some form of *history* must be kept from the past computations.

Theorems 5.6 and 5.7 provide a semantics, based on S-3-unsatisfiability and on S-1-satisfiability respectively, for each step of the algorithms for satisfiability based on the resolution principle and on enumeration of truth assignments. These two theorems also provide the basis for straightforward algorithms for checking S-3-unsatisfiability and S-1-satisfiability. Simply by generating the CNF formula T^m we can determine at the same time S-3-unsatisfiability and S-1-satisfiability of the formula T simply by checking whether T^m contains $\square$ or T^m is equal to Ω, in the same way this can be done through the generation of the set of formulae Φ^m. Moreover, both algorithms are incremental: if for a given S and T is still S-1-unsatisfiable and S-3-satisfiable and we take into account the set $S' = S \cup \{a_{m+1}\}$, then, in order to compute T^{m+1}, we only need T^m; in the same way, in order to compute Φ^{m+1} we only need Φ^m. These two algorithms fulfill our first two desiderata, but unfortunately they fail to accomplish the third one, because both algorithms use an amount of space which grows exponentially with the size of S.

This is particularly evident in the Davis–Putnam algorithm. In fact, we can perform both checks using a polynomially-bounded amount of space if we generate every combination $\delta = \{c_1, \ldots, c_m\}$, where each c_i $(1 \leqslant i \leqslant m)$ is either a_i or $\neg a_i$, one at the time. S-1-satisfiability is proven when we find a δ such that $simplify(T, \delta) = \Omega$. On the other hand, S-3-unsatisfiability is proven if for all δ we have that $simplify(T, \delta)$ contains the clause $\square$. This procedure fulfills the first and the third desiderata, but fails to fulfill the second one, since when we take into account a set $S' = S \cup \{a_{m+1}\}$ we are not able to exploit the previous computations and we must generate each combination $\{c_1, \ldots, c_m, c_{m+1}\}$.

Let us summarize all the above results and considerations about the relationships existing between S-satisfiability and classic algorithms for satisfiability. If the search tree of the resolution [Davis–Putnam] algorithm is visited breadth-first, resolving [splitting] on the letters of S in any order, then S-satisfiability "is the same as" satisfiability. Anyway in common implementations of resolution or Davis–Putnam methods the search tree is visited depth-first, using polynomial space. Therefore the possibility of using specialized heuristics—developed for selecting variables to resolve or split on—for making an appropriate choice of the set S is limited.

A further possibility for determining S-1- and S-3-satisfiability is to reduce T to T_S^1 and T_S^3, respectively, as sketched in the previous subsection, and check their 2-satisfiability.

These reductions may simplify the computations, but they force us to work on two different formulae, thus making difficult the integration of the checks. Furthermore, it is not clear how to make these computations incremental, since the reduced formulae T_S^3 and T_S^1 vary with the set S.

Our analysis has shown that all of the above algorithms fail to fulfill all of the desiderata, hence, we now develop an enumeration algorithm for performing the satisfiability check based on the following properties.

```
Algorithm S-3-unsat/S-1-sat;
Input an alphabet L, a subset S = {a_1, ..., a_m} of L, a CNF formula T
      and an integer j_in such that 0 ⩽ j_in ⩽ (2^(m-1) − 1);
Output true, iff T is S-1-satisfiable, false iff T is S-3-unsatisfiable,
      an integer j_out such that 0 ⩽ j_out ⩽ (2^m − 1) otherwise;
begin
  if there is a clause of T that does not contain at least one literal of S
  then S-1-unsat := true (* then T cannot be S-1-satisfiable *)
  else S-1-unsat := false; (* else T could be S-1-satisfiable *)
  S-1-sat := false;     (* S-1-sat is false until we prove S-1-satisfiability *)
  S-3-unsat := true;    (* S-3-unsat is true until we prove S-3-satisfiability *)
  exit := false;    (* this flag is used to control the loop *)
  j := 0;
  while (j < 2^m) and (not exit)
  do begin
       let d_1 ··· d_m be the binary coding of j; (* d_m is the least significant bit *)
       for i := 1 to m do   (* computation of a combination of literals *)
           if d_i = 1
           then c_i := a_i
           else c_i := ¬a_i;
       if (j ⩾ 2 * j_in) and (S-3-unsat)
       then if T ⊭^3 (c_1 ∨ ··· ∨ c_m)
            then begin
                   S-3-unsat := false;
                   j_out := j
                   if S-1-unsat (* T is both S-3-satisfiableand S-1-unsatisfiable *)
                   then exit := true; (* hence we exit the loop *)
                 end;
       if not S-1-unsat
       then if simplify(T; {c_1, ..., c_m}) contains no clauses
            then begin
                   S-1-sat := true; (* T is S-1-satisfiable *)
                   exit := true    (* hence we exit the loop *)
                 end;
       j := j + 1
     end;  (* while *)
  if S-1-sat then return true;
  if S-3-unsat then return false
              else return j_out
end;
```

Fig. 1. The algorithm S-3-unsat/S-1-sat.

Theorem 5.8 (Testing S-3-satisfiability). *Let S be the set $\{a_1, \ldots, a_m\}$. T is S-3-unsatisfiable iff $T \models^3 (a_1 \wedge \neg a_1) \vee \cdots \vee (a_m \wedge \neg a_m)$ holds, or equivalently $T \models^3 (c_1 \vee \cdots \vee c_m)$ holds for any combination $\{c_1, \ldots, c_m\}$, where each c_i $(1 \leqslant i \leqslant m)$ is either a_i or $\neg a_i$.*

Theorem 5.9 (Testing S-1-satisfiability). *Let S be the set $\{a_1, \ldots, a_m\}$. T is S-1-satisfiable iff there exists a set $\alpha = \{c_1, \ldots, c_m\}$, where each c_i $(1 \leqslant i \leqslant m)$ is either a_i or $\neg a_i$, such that simplify(T, α) is empty.*

The basic idea of the algorithm is that, given any set $S = \{a_1, \ldots, a_m\}$, at any step we generate a combination $\alpha = \{c_1, \ldots, c_m\}$ of literals such that each c_i $(1 \leqslant i \leqslant m)$ is either a_i or $\neg a_i$. To generate all the combinations $\{c_1, \ldots, c_m\}$ we use the binary coding of an integer j varying between 0 and $2^m - 1$. If *simplify*(T, α) is empty then T is S-1-satisfiable. Conversely, if $T \not\models^3 (c_1 \vee \cdots \vee c_m)$, then T is S-3-satisfiable. In the first case we have a definite answer, while in the second case we can stop our check of S-3-unsatisfiability. Notice that, if $\{c_1, \ldots, c_m\}$ is the first combination such that $T \not\models^3 (c_1 \vee \cdots \vee c_m)$ holds, then for any other combination $\{g_1, \ldots, g_m\}$ previously generated, $T \models^3 (g_1 \vee \cdots \vee g_m)$ holds. If $S' = S \cup \{a_{m+1}\}$ is the next set that we consider, we already know that $T \models^3 (g_1 \vee \cdots \vee g_m \vee \neg a_{m+1})$ and $T \models^3 (g_1 \vee \cdots \vee g_m \vee a_{m+1})$. Hence, in order to prove S'-3-satisfiability, we can start to generate combinations from $\{c_1, \ldots, c_m, \neg a_{m+1}\}$. Therefore, the combination $\{c_1, \ldots, c_m\}$ represents the history of our computation and is provided as an output through an integer j_{out}, which will be used as an input of the next call of the algorithm. In the same way, we suppose to have the integer j_{in} as input, representing the history of the previous call. Notice that if every clause of T does not contain at least one literal of S, then T is clearly S-1-unsatisfiable, therefore we do not need to generate its simplifications.

The algorithm is reported in Fig. 1. In this algorithm we assume that the set S is incremented of a single unit at any step, its generalization is obvious.

We notice that, if we restrict the algorithm to check S-3-unsatisfiability and we extend S through successive calls to the algorithm, then the number of checks of 3-entailment is less than or equal to $2^{|L|}$.

All the results presented in this section will be proved in Appendix A.2.

6. Approximate reasoning and non-omniscient agents

This section is an attempt at formalizing the form of approximate propositional reasoning introduced in Section 4 by means of a modal logic of knowledge. In the remaining Sections 7 and 8 we provide further applications of the approximation presented in Section 3.

Modal logics are probably the most widely used formalism to represent propositional attitudes, such as *knowledge* and *belief*, held by agents. For a detailed account of this usage of modal logics we refer to Hintikka's work [43].

One of the best-known shortcomings of this representation of agents is that agents represented through modal theories are ascribed the capability to infer any logical consequence of their knowledge (or belief). In other words, we make the unrealistic assumption that agents are capable of performing extremely complex inferences. This drawback

is known as the *logical omniscience problem* (see [43]).

Technically, logical omniscience is a consequence of the *possible-worlds* semantics commonly used [43, 51] as the semantics for logics of knowledge and belief. Generalizations of the possible-worlds semantics have been proposed to overcome the logical omniscience problem (for a detailed survey see [42]).

In the following we assume that the reader has some familiarity with modal logics and its best-known systems. We only want to remind that modal systems can be characterized either by a set of axiom schemata or by the corresponding constraints on the accessibility relation. A detailed description of the various modal systems can be found in the book by Chellas [16].

In this section we present a very general system, where "approximate" knowledge can be explicitly represented and used; this system is also compared with some of the formalisms presented in the literature.

The main idea underlying the system consists in providing language constructs for representing the kind of approximation implicit in the entailment relations $\models_S^3$ and $\models_S^1$ defined in Section 4. The system consists of two families of modal operators related to the notion of S-interpretation, where the elements of the first family are denoted as $\Box_S^3$ and the elements of the second family as $\Box_S^1$. Formulae are built using the usual connectives and the two sets of modal operators $\Box_S^3$ and $\Box_S^1$ for any $S \subseteq L$.

Formulae of the language are built over a set of literals L^* using the binary connectives $\wedge$ and $\vee$, the modal operators $\Box_S^3$ and $\Box_S^1$, their negation $\neg\Box_S^3$ and $\neg\Box_S^1$ plus parentheses. In the following, we refer to formulae in this form as *Modal Negation Normal Form* (MNNF). This restriction does not cause any loss of generality since any formula can be transformed into an equivalent one in MNNF by substituting implication $\rightarrow$ and by pushing negation inside the formulae, by means of the rules presented in Section 4 and the following rules:

$$(\Box\alpha)^* \mapsto \Box\alpha^*;$$
$$(\neg\Box\alpha)^* \mapsto \neg\Box\alpha^*.$$

When we discuss the semantics of general formulae we implicitly refer to their MNNF equivalent formula. Formulae not containing the modal operator are called simply propositional.

The semantics is a generalization of the classical possible-worlds semantics. A model is a triple $M = (Sit, R, V)$, where Sit is a set of situations, R is an accessibility relation, and V a valuation, which maps any situation into an unrestricted truth assignment $(V : Sit \rightarrow (L^* \rightarrow \{0,1\}))$. We denote with $\mathcal{W}(Sit)$ the set of situations $s \in Sit$ such that s is also a possible world, i.e. $V(s)$ is a 2-interpretation. Similarly, we denote with $S\text{-}3(Sit)$ and $S\text{-}1(Sit)$ the sets of situations that can be interpreted as S-3- and S-1-interpretations, respectively. The semantics is defined as follows (γ is a propositional formula):

- $\mathcal{M}, s \models \gamma$ iff $V(s)(\gamma) = 1$;
- $\mathcal{M}, s \models \Box_S^3\alpha$ iff $\forall t \in S\text{-}3(Sit)sRt$ implies $\mathcal{M}, t \models \alpha$;
- $\mathcal{M}, s \models \neg\Box_S^3\alpha$ iff $\exists t \in S\text{-}3(Sit)sRt$ and $\mathcal{M}, t \not\models \alpha$;
- $\mathcal{M}, s \models \Box_S^1\alpha$ iff $\forall t \in S\text{-}1(Sit)sRt$ implies $\mathcal{M}, t \models \alpha$;
- $\mathcal{M}, s \models \neg\Box_S^1\alpha$ iff $\exists t \in S\text{-}1(Sit)sRt$ and $\mathcal{M}, t \not\models \alpha$;

A formula α is valid, written $\models \alpha$, if α is true at every possible world $w \in \mathcal{W}(Sit)$ of every model $\mathcal{M} = (Sit, R, V)$. A formula α is satisfiable if there is a model $\mathcal{M} = (Sit, R, V)$ and a possible world $w \in \mathcal{W}(Sit)$ s.t. $\mathcal{M}, w \models \alpha$.

We notice that negation of modal formulae is defined in a classical fashion. In fact both $\Box_S^3 \alpha \wedge \neg \Box_S^3 \alpha$ and $\Box_S^1 \alpha \wedge \neg \Box_S^1 \alpha$ are unsatisfiable.

A minimal requirement for the system is its ability to represent the entailment relations $\models_S^1$ and $\models_S^3$ via the modal operators $\Box_S^1$ and $\Box_S^3$. This is in fact possible, as proven by the following result (α and γ are propositional formulae):

Theorem 6.1 (Modal validity and S-3-entailment). ($\models \Box_S^3 \alpha \rightarrow \Box_S^3 \gamma$) *iff* ($\Box_S^3 \alpha \wedge \neg \Box_S^3 \gamma$ *is unsatisfiable*) *iff* ($\alpha \models_S^3 \gamma$).

Theorem 6.2 (Modal validity and S-1-entailment). ($\models \Box_S^1 \alpha \rightarrow \Box_S^1 \gamma$) *iff* ($\Box_S^1 \alpha \wedge \neg \Box_S^1 \gamma$ *is unsatisfiable*) *iff* ($\alpha \models_S^1 \gamma$).

Hence, the proposed language can represent the approximate entailment relations. In order to make our system comparable with the modal system $\mathcal{S}5$ (see [16]), which is generally considered as an appropriate formalization of the notion of knowledge (see [43]), in the following we assume that the accessibility relation R is reflexive, transitive and euclidean.

It is now interesting to check whether the schemata defining the system $\mathcal{S}5$ are valid for these new operators, in order to show their adequacy to represent resource-bounded agents.

The system $\mathcal{S}5$ is characterized by the usual rules and axiom schemata of the propositional calculus plus the inference rule (necessitation):

(Nec) $\models \alpha$ implies $\models \boldsymbol{K}\alpha$.

and the axiom schemata:

(K) $\boldsymbol{K}(\alpha \rightarrow \beta) \rightarrow (\boldsymbol{K}\alpha \rightarrow \boldsymbol{K}\beta)$.

(T) $\boldsymbol{K}\alpha \rightarrow \alpha$.

(4) $\boldsymbol{K}\alpha \rightarrow \boldsymbol{K}\boldsymbol{K}\alpha$.

(5) $\neg \boldsymbol{K}\alpha \rightarrow \boldsymbol{K}\neg \boldsymbol{K}\alpha$.

We now analyze which of these schemata are valid in our semantics when we replace $\boldsymbol{K}$ with $\Box_S^1$. The result is the following:

(Nec) $\models \alpha$ does not imply $\models \Box_S^1 \alpha$.

(K) $\models \Box_S^1(\alpha \rightarrow \beta) \rightarrow (\Box_S^1 \alpha \rightarrow \Box_S^1 \beta)$.

(T) $\not\models \Box_S^1 \alpha \rightarrow \alpha$.

(4) $\models \Box_S^1 \alpha \rightarrow \Box_S^1 \Box_S^1 \alpha$.

(5) $\models \neg \Box_S^1 \alpha \rightarrow \Box_S^1 \neg \Box_S^1 \alpha$.

The validity of the schemata 4 and 5 is a straightforward consequence of the properties of the accessibility relation, while the schema K follows from the semantic definition of $\Box_S^1$. We now show counterexamples for the properties which do not hold.

(Nec) Let $\alpha = q \vee \neg q$, $S = \emptyset$, $\mathcal{M} = (Sit, R, V)$, $Sit = \{s_1, s_2\}$, $R = \{(s_1, s_1), (s_1, s_2), (s_2, s_1), (s_2, s_2)\}$, $V(s_1)(q) = 1$ and $V(s_1)(\neg q) = V(s_2)(q) = V(s_2)(\neg q) = 0$. Notice that $\mathcal{W}(Sit) = \{s_1\}$. We have that $\models \alpha$ holds but $\models \Box_S^1 \alpha$ does not hold, in fact $\mathcal{M}, s_1 \not\models \Box_S^1 \alpha$.

(T) Let $S = \emptyset$, $\mathcal{M} = (Sit, R, V)$, $Sit = \{s_1\}$, $R = \{(s_1, s_1)\}$, $V(s_1)(\neg p) = 1$ and $V(s_1)(p) = 0$. Notice that $\mathcal{W}(Sit) = \{s_1\}$. $\Box_S^1 p \rightarrow p$ is valid iff $\forall N \; \forall w \in \mathcal{W}(Sit)$ $N, w \models \neg \Box_S^1 p \vee N, w \models p$. Since by instantiating N to $\mathcal{M}$ and w to s_1 we obtain that $\mathcal{M}, s_1 \not\models \neg \Box_S^1 p$ and $\mathcal{M}, s_1 \not\models p$ then it is not the case that $\Box_S^1 \alpha \rightarrow \alpha$ is valid.

We have shown that both the rule of necessitation and the axiom schema T do not hold in general. As a consequence we can use $\Box_S^1$ to model an agent capable of performing *at least* every sound inference, because its knowledge is closed under modus ponens (the K schema), nevertheless, the agent can do some inference which is not sound, in fact the T schema does not hold. Since both schemata 4 and 5 are valid, it follows that agents modeled in our system are fully introspective. Even if the necessitation rule and the T schema do not hold in general they are valid whenever $letters(\alpha) \subseteq S$, so they still hold for a subset of the language.

We now analyze which schemata are valid in our semantics when we replace $\boldsymbol{K}$ with $\Box_S^3$.

(Nec) $\models \alpha$ implies $\models \Box_S^3 \alpha$.

(K) $\not\models \Box_S^3(\alpha \rightarrow \beta) \rightarrow (\Box_S^3 \alpha \rightarrow \Box_S^3 \beta)$.

(T) $\models \Box_S^3 \alpha \rightarrow \alpha$.

(4) $\models \Box_S^3 \alpha \rightarrow \Box_S^3 \Box_S^3 \alpha$.

(5) $\models \neg \Box_S^3 \alpha \rightarrow \Box_S^3 \neg \Box_S^3 \alpha$.

Again, the schemata 4 and 5 are a straightforward consequence of the properties of the accessibility relation, while the schema T and the rule Nec follow from the semantic definition of $\Box_S^3$. We now show a counterexample for the property K. Let $S = \emptyset$, $\mathcal{M} = (Sit, R, V)$, $Sit = \{s_1, s_2\}$, $R = \{(s_1, s_1), (s_1, s_2), (s_2, s_1), (s_2, s_2)\}$, $V(s_1)(p) = V(s_1)(q) = V(s_2)(p) = V(s_2)(\neg p) = V(s_2)(\neg q) = 1$ and $V(s_1)(\neg p) = V(s_1)(\neg q) = V(s_2)(q) = 0$. Notice that $\mathcal{W}(Sit) = \{s_1\}$. We have that $\mathcal{M}, s_1 \models \Box_S^3(p \rightarrow q)$ and $\mathcal{M}, s_1 \models \Box_S^3 p$ but $\mathcal{M}, s_1 \models \Box_S^3 q$ does not hold.

Even this set of modal operators does not satisfy all the rules and axiom schemata of $\mathcal{S}5$. In this case the only property which is not satisfied is property K, that is the closure of the knowledge under *modus ponens*. This implies that an agent modeled with these operators is not committed to full logical omniscience. Its deductive capabilities are limited, but its inferences are always sound, as witnessed by the validity of the T schema. Again properties 4 and 5 continue to hold thus providing the agent with perfect introspection. Since $\Box_S^3$ is an approximation of $\boldsymbol{K}$, it is clear that whenever the set S is equal to L the two operators coincide and therefore $\Box_L^3$ will satisfy exactly the same properties of $\boldsymbol{K}$, including the K schema. But this schema is also satisfied under weaker

conditions, if α is a propositional formula we have that $\Box_S^3(\alpha \rightarrow \beta) \rightarrow (\Box_S^3\alpha \rightarrow \Box_S^3\beta)$ is valid if and only if $\alpha \wedge \neg\alpha$ is S-3-unsatisfiable.

There are other interesting properties which involve different sets S, for example we have that

Theorem 6.3 (Monotonicity for $\Box_S^3$). *Let* $S \subseteq S' \subseteq L$. $\models \Box_S^3\alpha \rightarrow \Box_{S'}^3\alpha$.

We claim that the two sets of operators $\Box_S^3$ and $\Box_S^1$ account for a non-ideal reasoner with a limited amount of resources. The first set of operators accounts for a skeptical reasoner, while the second set accounts for a credulous one. We can now model an agent's reasoning capabilities avoiding the logical omniscience assumption and providing a set of semantically motivated restrictions to its deductive capabilities. In particular, we claim that our system can be used for modeling the interaction of several non-ideal agents, each having its own attitude—either credulous or skeptical—and competence, characterized by the set S.

It is commonplace that knowledge about the structure of a knowledge base can be very useful in the choice of an efficient query-answering strategy. In general, the knowledge engineer who creates a knowledge base is aware of some knowledge about its structure.

If we can design a language well suited to represent this kind of knowledge, then we can take advantage of it when the system is asked to answer queries. While the importance to represent this information is generally acknowledged, very little has been done in practice in order to achieve this goal.

We show how the proposed language can be (sometimes) useful to represent information about the knowledge base. As an example, the knowledge engineer of the knowledge base T, which contains knowledge about animals, may know that, in order to decide whether cows have molar teeth, it is sufficient to take into account only the properties of herbivores. This information can be modeled by a formula like:

$$[T \rightarrow (\mathit{cow} \rightarrow \mathit{molar\text{-}teeth})] \rightarrow [\Box_S^3 T \rightarrow \Box_S^3(\mathit{cow} \rightarrow \mathit{molar\text{-}teeth})]$$

for a suitable set S of propositional letters, for example all letters expressing properties of the herbivores. The intuitive reading of the above formula is: if it holds that cows have molar teeth, then we can prove it using only S-3-entailment, for a fixed S. In the query-answering process this meta-knowledge should be automatically used to perform an "intelligent" choice of the set S which guarantees an high degree of confidence (cf. end of Section 4).

The specification of actual procedures for taking advantage of structural knowledge represented in this meta-language deserves further investigation.

Several other systems capable of representing non-omniscient agents have been presented in the literature. We now briefly compare our system with some of the best-known formalisms.

In Levesque's system [54] the semantics of the modal operator of implicit belief $\boldsymbol{L}$ is defined in terms of possible worlds. Conversely, the modal operator of explicit belief $\boldsymbol{B}$ is defined in terms of situations. Both operators can be represented in our semantics and with our operators, in fact, $\boldsymbol{L}$ is equivalent to $\Box_S^3$ or $\Box_S^1$ when $S = L$ and $\boldsymbol{B}$ is equivalent to $\Box_S^3$ when $S = \emptyset$. Hence, $\Box_S^3\alpha$ has the intuitive reading of "α can be made explicit by

reasoning only on letters in S".

Lakemeyer [53] has extended Levesque's system, allowing a restricted form of nesting of operators. However, his semantics, which relies on two distinct accessibility relations R and $\overline{R}$ for interpreting negated formulae, is very different from ours, thus making difficult any comparison.

A very interesting proposal has been presented by Fagin and Halpern in [33]. In that paper they introduce a new propositional attitude, in addition to knowledge and belief: the attitude of awareness. This new modality should act as a kind of filter on the consequences that can be drawn. In their system, truth in a world is defined in terms of the relation $\models^{\Psi}$, where $\Psi \subseteq L$ is a set of propositional letters and the agent is aware only of them. The intended meaning of $\models^{\Psi}$ is to restrict the attention only to letters in Ψ, while letters in $L \setminus \Psi$ are ignored. Our notion of S-1-interpretation is exactly in the same spirit. However, in Fagin and Halpern's system there is nothing close in spirit to the notion of S-3-interpretation.

In a more recent work [34], Fagin, Halpern and Vardi present a different system which does not commit to the logical omniscience assumption. The presented system clarifies the reason why most of the non-classical semantics are not committed to logical omniscience. The main reasons are the impossibility of distinguishing either between coherent and incoherent worlds or between complete and incomplete ones. It is exactly the possibility of discerning between the various degrees of incoherence and incompleteness which has led us to the definition of our system. For example, the effect of the operator $\Box_S^3$ is exactly to select only complete situations which can be only partially incoherent, i.e. can be incoherent only in the interpretation of the propositions in $L \setminus S$. Analogous is the effect of $\Box_S^1$ which selects only coherent situations being partially incomplete.

All the results presented in this section will be proved in Appendix A.3.

The results of this section have been published in a preliminary form in [12].

7. Concept description languages

In this section we generalize the technique introduced in Section 4 to deal with (fragments of) first-order logic. In particular, we take into account the so-called *concept description logics*, also known as terminological logics, that are abstractions for several languages in computer science and artificial intelligence. The importance of concept languages in data modeling [7], object-oriented databases [3] and logic programming [1] has been stressed by several authors.

The computational complexity of concept languages has been extensively studied by many researchers (see [25, 27, 59, 68]). In particular, it has been shown that reasoning is polynomially intractable in many interesting cases, and that tractability depends on small variations of the expressiveness of the language. A big effort has been spent in the design of "maximally polynomial languages", i.e. polynomial languages such that no expressiveness can be added without losing tractability.

Other researchers (see [58, 61]) proposed incomplete or unsound reasoning systems based on non-standard semantics in order to simplify reasoning tasks for concept languages. Patel-Schneider in [61] presented a polynomially tractable terminological logic

based on a 4-valued semantics. Kautz and Selman proposed in [69] a syntactical simplification of concepts in order to make inference computationally more tractable, by the definition of sound approximations and complete approximations.

In this paper we focus our attention on the languages belonging to the $\mathcal{AL}$-family (see [68] for an overview on the family) and in particular on $\mathcal{ALE}$ and $\mathcal{ALC}$. As shown in [26], these languages are representative of a wide class of concept languages.

The structure of this section is as follows: in the following subsection we recall the basic notions on concept languages, while in Section 7.2 we introduce multivalued logics for approximation in first-order languages. In Sections 7.3 and 7.4 we demonstrate our approximation method for the two languages $\mathcal{ALE}$ and $\mathcal{ALC}$, respectively.

7.1. Concept languages

In this subsection we briefly summarize the syntax and semantics of concept description languages.

Concept languages are decidable sublanguages of predicate logic, designed for dealing with *concepts*. A concept is a monadic predicate that can be built up of two kinds of symbols, primitive concepts and roles. Primitive entities can be combined by various language constructors yielding complex concepts. A primitive concept is simply a symbol, like female, that is used to represent a class of individuals all having a common property, like being female. A primitive role is a symbol, like friend, that is used to represent a binary relation among individuals, like friendship. A typical example of a language constructor is the universal quantification $\forall$. The symbol $\forall$friend.female is a composite concept that represents the set of individuals having all the friends being female. Other typical language constructors are the existential quantification $\exists$ and the boolean connectives $\sqcap$, $\sqcup$, $\neg$. Another example of composite concept is $\exists$friend.$\neg$female, that represents the set of individuals having at least one friend being not female.

In this section we focus our attention on the two languages $\mathcal{ALE}$ and $\mathcal{ALC}$ belonging to the $\mathcal{AL}$-family (see [68]).

Throughout this section we denote primitive concepts with the letters A and B, concepts (either primitive or composite) with the letters C and D and primitive roles with the letter R.

The syntax of the language $\mathcal{ALC}$ is the following:

$$C, D \longrightarrow \top \mid \bot \mid A \mid \neg A \mid C \sqcap D \mid C \sqcup D \mid \forall R.C \mid \exists R.C$$

The special symbols $\top$ and $\bot$ stand for the universal concept and the empty concept, respectively. All the individuals belong to $\top$, while no individual belongs to $\bot$. Although the standard syntax of $\mathcal{ALC}$ allows negation in front of any concept, due to the presence of disjunction we don't lose any generality by allowing negation only in front of primitive concepts. Examples of well-formed $\mathcal{ALC}$ concepts are:

$$(\exists R.A) \sqcap (\exists R.(\neg A \sqcap \forall R.\exists R.A)),$$

$$\exists R.\exists R.\forall R.(A \sqcup B).$$

$\mathcal{ALE}$ is another interesting language, whose syntax is a restriction of that of $\mathcal{ALC}$. In particular the construct $C \sqcup D$ is not allowed in $\mathcal{ALE}$, therefore the first of the two $\mathcal{ALC}$ concepts above is also an $\mathcal{ALE}$ concept, while the second is not.

The semantics of concept languages is usually given by defining a domain of interpretation U, by assigning to every primitive concept a subset of U and to every primitive role a subset of $U \times U$ and by defining a rule for each constructor. For the purpose of our work we prefer to define the semantics of $\mathcal{ALE}$ and $\mathcal{ALC}$ by translating any concept C into a first-order sentence $\Gamma(C)$ and interpreting it with the classical semantics of first-order logic. As an example, the $\mathcal{ALC}$ concept $D \equiv (B \sqcap \forall R.A) \sqcup \exists R.\neg B$ is translated into the first-order formula

$$(B(a) \wedge \forall x R(a,x) \rightarrow A(x)) \vee (\exists y R(a,y) \wedge \neg B(y)),$$

where a is a constant symbol. In general the translation from a concept C into the corresponding first-order formula is obtained by applying the following rewriting rules to C:

$$\begin{aligned}
&\Gamma(C) \mapsto \Phi(C,a),\\
&\Phi(\top,x) \mapsto t,\\
&\Phi(\bot,x) \mapsto f,\\
&\Phi(A,x) \mapsto A(x),\\
&\Phi(\neg A,x) \mapsto \neg A(x),\\
&\Phi(C \sqcap D,x) \mapsto \Phi(C,x) \wedge \Phi(D,x),\\
&\Phi(C \sqcup D,x) \mapsto \Phi(C,x) \vee \Phi(D,x),\\
&\Phi(\forall R.C,x) \mapsto \forall y R(x,y) \rightarrow \Phi(C,y),\\
&\Phi(\exists R.C,x) \mapsto \exists y R(x,y) \wedge \Phi(C,y),
\end{aligned}$$

where a is a constant symbol, y is a variable symbol and x is either the constant symbol a or a variable symbol. We assume that the new variables introduced by the last two rules are fresh. A concept C is *satisfiable* iff its translation $\Gamma(C)$ is a satisfiable sentence, *unsatisfiable* otherwise. As an example, the concept $\forall R.A$ is satisfiable, while the concept $(\forall R.A) \sqcap \exists R.\neg A$ is unsatisfiable. It is immediate to show that this semantics is equivalent to the more standard semantics for concept languages as found, for example, in [68].

In this section we are interested in analyzing satisfiability problems, although the computational complexity of other reasoning tasks like subsumption and instance checking has been extensively studied in the literature. The complexity analysis has shown that satisfiability checking is PSPACE-complete for $\mathcal{ALC}$ [68] and co-NP-complete for $\mathcal{ALE}$ [25].

7.2. *First-order generalization of S-interpretations*

In this subsection we define a generalized notion of interpretation of first-order formulae, in particular extending to a first-order semantics the definitions of S-3- and

S-1-interpretations introduced in Section 4. This new notion will be used in the following for defining approximations methods for satisfiability of $\mathcal{ALE}$ and $\mathcal{ALC}$ concepts.

In order to simplify our definitions, we consider fixed domains of interpretation for the models. In particular we only consider Herbrand models. To this end, we assume that all sentences are in Skolem Normal Form. Since we are interested only in studying the satisfiability of sentences, the Herbrand theorem ensures that we are not losing any generality.

We recall that the Herbrand Universe U_h of a formula ϕ is the set of terms that can be built using the constant and function symbols occurring in ϕ (if no constants occur in ϕ include one in U_h). The Herbrand Base B_h of ϕ is the set of ground atoms built applying predicate symbols occurring in ϕ to terms of the Herbrand Universe. A Herbrand interpretation of the signature $\mathcal{L}$ used in ϕ is a pair $I = \langle I^+, I^- \rangle$ of subsets of the Herbrand Base B_h which partition B_h. In other words I is built using two rules:

Rule 1. $I^+ \cup I^- = B_h$.

Rule 2. $I^+ \cap I^- = \emptyset$.

The subset I^+ represents the set of atoms α of B_h which are true in I, while I^- represents the set of atoms α of B_h whose negation is true in I. Since there are only two possibilities for an atom, namely it may belong either to I^+ or to I^-, we are entitled to call the partition I a 2-*interpretation* of $\mathcal{L}$. A 2-interpretation I satisfies an atom α ($I \models \alpha$) iff $\alpha \in I^+$ and satisfies a negated atom $\neg\alpha$ ($I \models \neg\alpha$) iff $\alpha \in I^-$. The satisfaction of complex sentences is defined using the standard rules for disjunction, conjunction and universal quantification:

- $\vee$-*rule*: $I \models \alpha \vee \beta$ iff $I \models \alpha$ or $I \models \beta$;
- $\wedge$-*rule*: $I \models \alpha \wedge \beta$ iff $I \models \alpha$ and $I \models \beta$;
- $\forall$-*rule*: $I \models \forall x.\gamma(x)$ iff $I \models \gamma(t)$ for all $t \in U_h$.

We are not interested in rules for the interpretation of complex negated formulae, since it is always possible to "push" the negation in front of atoms using a variant of the rewriting rules presented in Section 4.

The main idea of our semantic notion of approximation is to define interpretation of formulae without using one of Rules 1 or 2. This idea is a generalization of the idea presented in Section 4. In fact, loosely speaking, the Herbrand Base of a propositional formula is the set of the boolean variables occurring in it, therefore the Definition 2.1 of 3-interpretation given by Levesque corresponds to keep Rule 1 but not Rule 2.

We use the symbol $\mathcal{L}$ to denote a signature. We define a 3-*interpretation* of $\mathcal{L}$ to be a pair $I = \langle I^+, I^- \rangle$ of subsets of B_h such that Rule 1 holds and Rule 2 may not hold. Interpretation of complex formulae is defined by means of the $\vee$-rule, $\wedge$-rule and $\forall$-rule.

The notions of satisfiability and entailment are defined as in the propositional case.

Definition 4.1 of S-3-interpretation, presented in Section 4 makes use of Rule 2 in a restricted way. We now give the definition that applies to first-order formulae. Let S be a subset—even not proper—of the Herbrand Base B_h.

Definition 7.1 (*S-3-interpretation of first-order formulae*). An S-3-interpretation of $\mathcal{L}$ is a pair $I = \langle I^+, I^- \rangle$ where both I^+ and I^- are subsets of B_h, such that

Rule 1. $I^+ \cup I^- = B_h$.

Rule 2. $I^+ \cap I^- \cap S = \emptyset$.

Intuitively, an S-3-interpretation is a 2-interpretation of the atoms of S, while it is a 3-interpretation of the remaining atoms. The three possibilities for each atom α of B_h are the following:

(1) $\alpha \in I^+$ and $\alpha \notin I^-$;
(2) $\alpha \notin I^+$ and $\alpha \in I^-$;
(3) $\alpha \in I^+$ and $\alpha \in I^-$ [only if $\alpha \in L \setminus S$].

Notice that, for any S, a 2-interpretation is always an S-3-interpretation, while the latter is always a 3-interpretation. Satisfaction of complex sentences is defined by means of the $\vee$-rule, $\wedge$-rule and $\forall$-rule. Notice also that if $S \subseteq S' \subseteq B_h$, then S-3-unsatisfiability of a formula always implies its S'-3-unsatisfiability, hence its 2-unsatisfiability.

Using this characteristic of S-3-satisfiability we can approximate the satisfiability problem for a formula by means of a sequence $\langle S_0 \subset S_1 \subset \cdots \subset S_n = B_h \rangle$ of sets. In the following we show actual examples of the choice of this kind of sequences in the context of the languages $\mathcal{ALE}$ and $\mathcal{ALC}$.

We now introduce the definition of S-1-interpretation of first-order formulae, which is dual to S-3-interpretation and modifies Rule 1 while keeping Rule 2.

Definition 7.2 (*S-1-interpretation of first-order formulae*). An S-1-interpretation of $\mathcal{L}$ is a pair $I = \langle I^+, I^- \rangle$ where both I^+ and I^- are subsets of B_h, such that

Rule 1. $I^+ \cup I^- = S$.

Rule 2. $I^+ \cap I^- = \emptyset$.

An S-1-interpretation is a 2-interpretation as long as the atoms of S are concerned, while for all the atoms α not in S it holds that $\alpha \notin I^+$ and $\alpha \notin I^-$.

We define $Terms(S)$ to be the set of all the terms that can be generated by using the functions and constants which appear in atoms of S. Notice that this is in general a subset of the Herbrand Universe. We define S-1-interpretation by means of the $\vee$-rule, $\wedge$-rule and the following

- $\forall'$*-rule*: $I \models \forall x.\gamma(x)$ iff $I \models \gamma(t)$ for all $t \in Terms(S)$.

As we show in the following subsections, we are not interested in sets S containing all the ground instances of a predicate symbol. S-1-interpretation with the $\forall$-rule would be trivial, since no sentence can be satisfied. The intuition behind the $\forall'$-rule is that we are ignoring objects that are not in the intended domain of interpretation.

7.3. Approximation in $\mathcal{ALE}$

We now define a method for approximating the task of deciding satisfiability of an $\mathcal{ALE}$ concept. The method is based on a syntactic manipulation of concepts that

simplifies the task of checking their satisfiability. The syntactic manipulation is given a precise semantics in terms of S-1- and S-3-interpretations.

The method we are going to present is based on the idea of approximating an $\mathcal{ALE}$ concept by means of two sequences of "simpler" $\mathcal{ALE}$ concepts. There are two ways in which a concept can be simpler than another one: a concept can be approximated either by a *weaker* concept or by a *stronger* one. A concept D is weaker than C if it represents a class with weaker properties, i.e. a less specific class. On the other hand, stronger concepts represent more specific classes. Both kinds of approximated concepts carry interesting information. In fact, if we can prove that a weaker concept is unsatisfiable, then the unsatisfiability of C is also proved. Proving the satisfiability of a stronger concept implies the satisfiability of C. One of the two sequences defining the approximation contains only weaker concepts. It starts with a very rough approximation and is improved in a stepwise process, giving "stronger and stronger" approximations and eventually converging to the original concept. The second sequence is dual, and contains only stronger concepts.

As an example, let's consider an unsatisfiable $\mathcal{ALE}$ concept, called *dummy* in the following

$$(\exists \mathsf{friend.tall}) \sqcap$$
$$\forall \mathsf{friend.}((\forall \mathsf{friend.doctor}) \sqcap \exists \mathsf{friend.}\neg \mathsf{doctor}).$$

It denotes the (empty) set of the individuals having at least one friend tall and all the friends having both all the friends doctor and at least one friend who is not a doctor.

For obtaining concepts approximating *dummy* we syntactically simplify it by substituting complex subconcepts with simpler ones, where a subconcept D of an $\mathcal{ALE}$ concept C is a substring of C being an $\mathcal{ALE}$ concept. The *depth* of D is the number of universal quantifiers occurring in C and having D in their scope. For example, the depth of the subconcept $\exists \mathsf{friend.}\neg \mathsf{doctor}$ of *dummy* is 1, while the depth of $\exists \mathsf{friend.tall}$ is 0. We define the depth of a concept to be the maximum depth of its existentially quantified subconcepts. The depth of *dummy* is 1.

Using the notion of depth we define the sequence of weaker approximated concepts. The ith weaker concept is obtained by replacing every existentially quantified subconcept of depth greater or equal than i with the primitive concept $\top$. As an example, taking into account *dummy* we obtain the sequence of concepts:

- $\top \sqcap \forall \mathsf{friend.}((\forall \mathsf{friend.doctor}) \sqcap \top)$.
- $(\exists \mathsf{friend.tall}) \sqcap \forall \mathsf{friend.}((\forall \mathsf{friend.doctor}) \sqcap \top)$.
- *dummy*.

The elements of the sequence are denoted as $dummy_0^{\top}$, $dummy_1^{\top}$ and $dummy_2^{\top}$, respectively. Notice that the first two concepts are both satisfiable, while the third one is unsatisfiable. The sequence of the stronger concepts is obtained by substituting $\bot$ instead of $\top$. Its elements are denoted as $dummy_0^{\bot}$, $dummy_1^{\bot}$ and $dummy_2^{\bot}$, respectively.

Formally, we associate to an $\mathcal{ALE}$ concept C of depth n two distinct sequences $\sigma^{\top} = C_0^{\top}, \ldots, C_{n+1}^{\top}$ and $\sigma^{\bot} = C_0^{\bot}, \ldots, C_{n+1}^{\bot}$ of $\mathcal{ALE}$ concepts. For each i $(0 \leqslant i \leqslant n)$,

every concept $C_i^{\top}$ $[C_i^{\bot}]$ of the sequence $\sigma^{\top}$ $[\sigma^{\bot}]$ is obtained from C by substituting every existentially quantified subconcept of C which is in the scope of at least i universal quantifiers with the concept $\top$ $[\bot]$. Moreover $C_{n+1}^{\top} = C_{n+1}^{\bot} = C$. Notice that, for each i $(1 \leqslant i \leqslant n+1)$ the depth of $C_i^{\top}$ $[C_i^{\bot}]$ is strictly less than i.

The semantics of the approximation will be addressed later on, but at this point we would like to make some intuitive considerations on the simplified concepts. As we noticed in Section 7.1, proving the satisfiability of a concept C is equivalent to proving the existence of an object a having the property represented by the first-order formula $\Gamma(C)$. If existentially quantified subconcepts of depth 0 occur in C, then it is necessary to consider objects related to a. As an example, when we deal with the concept *dummy* we need to consider the existence of a friend of a being tall, because of the subconcept $\exists$friend.tall of C. Let's call c this friend. Deeper existentially quantified subconcepts may also contribute to the generation of objects that have to be considered. Continuing our example, the subconcept $\exists$friend.$\neg$doctor of *dummy* makes it necessary to take into account the existence of a friend of c—let's call it $f(c)$—not being a doctor. Intuitively, focusing on simplified concepts means ignoring the properties of objects which are "far away" from a. As an example, considering $dummy_1^{\top}$ frees us from taking into account the object $f(c)$, while when we consider $dummy_0^{\top}$ we don't even have to deal with the object c. This intuition will be clarified when we will analyze the semantics of the concepts in the approximating sequences.

We now give a couple of important properties of the sequences $\sigma^{\top}$ and $\sigma^{\bot}$, that are useful for defining our approximation schema.

Theorem 7.3 (Monotonicity of $\sigma^{\top}$ and $\sigma^{\bot}$). *For each i $(0 \leqslant i \leqslant n+1)$, if $C_i^{\top}$ is unsatisfiable then $C_j^{\top}$ is unsatisfiable for all $j \geqslant i$, hence C is unsatisfiable.*

For each i $(0 \leqslant i \leqslant n+1)$, if $C_i^{\bot}$ is satisfiable then $C_j^{\bot}$ is satisfiable for all $j \geqslant i$, hence C is satisfiable.

Observation 7.4 (Convergence of $\sigma^{\top}$ and $\sigma^{\bot}$). *If C is unsatisfiable, then there exists an i $(0 \leqslant i \leqslant n+1)$ such that $C_i^{\top}$ is unsatisfiable. If C is satisfiable, then there exists an i $(0 \leqslant i \leqslant n+1)$ such that $C_i^{\bot}$ is satisfiable.*

We notice that these results correspond to Theorem 4.4 and Observation 4.5 shown in Section 4. Theorem 7.3 and Observation 7.4 suggest a method for deciding in an incremental fashion the satisfiability of an $\mathcal{ALE}$ concept C. We may start by deciding the satisfiability of $C_0^{\top}$; if $C_0^{\top}$ is unsatisfiable, then by Theorem 7.3 we are guaranteed that C is unsatisfiable as well. Analogously, if $C_0^{\bot}$ is satisfiable, then we know that C is satisfiable. If neither of the two cases happens, then we decide the satisfiability of $C_1^{\top}$ and $C_1^{\bot}$, and so on. Clearly we have a definite answer as soon as we prove either unsatisfiability of some $C_i^{\top}$ or satisfiability of some $C_i^{\bot}$. Referring to the notation defined in Section 3, we approximate the set $\mathcal{D}$ of satisfiable $\mathcal{ALE}$ concepts of depth n by means of two sequences of sets $\langle \mathcal{D}_0, \mathcal{D}_1, \ldots, \mathcal{D}_{n+1} \rangle$ and $\langle \mathcal{D}^0, \mathcal{D}^1, \ldots, \mathcal{D}^{n+1} \rangle$, where $\mathcal{D}_i$ $[\mathcal{D}^i]$ is the set of $\mathcal{ALE}$ concepts C whose stronger [weaker] form $C_i^{\bot}$ $[C_i^{\top}]$ is satisfiable.

We now make some considerations on the computational cost of deciding satisfiability

of an $\mathcal{ALE}$ concept in such an incremental fashion. Donini et al. show in [25] that the problem of checking the satisfiability of an $\mathcal{ALE}$ concept whose depth is linear in its length is co-NP-complete. In the same paper it is shown that the satisfiability of any $\mathcal{ALE}$ concept having depth m can be checked in time proportional to $l \cdot 2^m$, where l is the length of the concept. In other words, the nesting of existential and universal quantifiers is the crucial measure of the complexity of satisfiability checking. If such a nesting is bounded by a constant, then satisfiability can be tested in linear time. This observation has the same importance of Theorem 4.6 presented in Section 4.

Our method for checking satisfiability of an $\mathcal{ALE}$ concept C considers simplified versions of C of increasing depth and may use existing algorithms for checking their satisfiability. The complexity of the whole method is $O(l^2 \cdot 2^m)$ even if satisfiability cannot be decided until the unsimplified concept C is taken into account. In the worst case the complexity of our method is therefore comparable to that of the existing algorithms. Since the problem is co-NP-complete, we don't expect any algorithm to be significantly better than ours in the worst case.

We are now interested in giving a clear semantics to our approximation schema. This is particularly important, since as we stressed earlier it is not always possible to obtain in a reasonable amount of time a definite answer to the problem of checking the satisfiability of an $\mathcal{ALE}$ concept. Therefore the meaning of each step of the approximation should be very clear, since in general we can afford only an approximate solution.

In order to give a semantic account to the approximations of $\mathcal{ALE}$ concepts, we are now going to use the notions of S-3- and S-1-interpretation introduced in Section 7.2 As we noticed earlier, S-3-interpretations lead to a complete (and in general unsound) definition of satisfiability. Analogously, Theorem 7.3 shows that satisfiability of the concepts of the sequence $\sigma^\top$ is complete and unsound with respect to the satisfiability of the original concept C. Our goal is to show that 2-satisfiability (or simply, satisfiability) of each concept $C_i^\top$ is equivalent to S_i-3-satisfiability of C for a suitable subset S_i of the Herbrand Base of the Skolem Normal Form $SNF(\Gamma(C))$ of $\Gamma(C)$. An analogous result will be obtained for a concept $C_i^\perp$ and S_i-1-satisfiability of C.

As we said in Section 7.1, the semantics of an $\mathcal{ALE}$ concept C can be defined in terms of a first-order formula $\Gamma(C)$. Since we are only interested in satisfiability properties, we take into account the Skolem Normal Form $SNF(\Gamma(C))$ of $\Gamma(C)$. The Herbrand Universe of $SNF(\Gamma(C))$ can be stratified in a very simple way by taking into account the increasing complexity of its terms. This stratification delivers another simple stratification on the Herbrand Base. More precisely, let U_h be the Herbrand Universe of $SNF(\Gamma(C))$. This universe can be stratified with the following policy: Let U_0 be $\{a\}$ and U_i be the set of all the terms which can be formed using a, Skolem constants of U_h and functions of U_h of arity strictly less than i. Clearly it holds that $U_0 \subseteq U_1 \subseteq \cdots \subseteq U_{n+1} = U_h$. Notice that a function of arity i comes from the skolemization of an existential quantifier of depth i. This stratification of the Herbrand Universe of $SNF(\Gamma(C))$ induces a stratification $S_0 \subseteq S_1 \subseteq \cdots \subseteq S_{n+1} = B_h$ on its Herbrand Base B_h defined in the following way:

$$S_i = \{A(t) \mid A \text{ is a primitive concept and } t \in U_i\} \cup \{R(t_1, t_2) \mid R \text{ is a role and } t_1, t_2 \in U_i\}.$$

A clarifying example is in order. Let's consider again the concept *dummy*

$$(\exists \text{friend.tall}) \sqcap$$
$$\forall \text{friend.}((\forall \text{friend.doctor}) \sqcap \exists \text{friend.}\neg \text{doctor})$$

and the related first-order formula $\Gamma(dummy)$, in which we use obvious abbreviations for the predicate symbols

$$(\exists x\, F(a,x) \wedge T(x)) \wedge \forall y\, F(a,y) \rightarrow$$
$$(\forall z(F(y,z) \rightarrow D(z)) \wedge (\exists u F(y,u) \wedge \neg D(u))).$$

The following formula is the Skolem Normal Form $SNF(\Gamma(dummy))$ of $\Gamma(dummy)$

$$F(a,c) \wedge T(c) \wedge \forall y(F(a,y) \rightarrow$$
$$(\forall z(F(y,z) \rightarrow D(z)) \wedge F(y,f(y)) \wedge \neg D(f(y)))).$$

In $SNF(\Gamma(dummy))$, c is a new constant symbol that replaces the variable x, while $f()$ is a new function symbol—having arity 1—replacing the variable u. The Herbrand Universe U_h of $SNF(\Gamma(dummy))$ is the set of all the terms that can be obtained from a, c and $f()$. The Universe U_h is stratified into the three sets $U_0 = \{a\}$, $U_1 = \{a, c\}$ and $U_2 = \{a, c, f(a), f(c), f(f(a)), \ldots\}$.

Notice that S_i turns out to be equal (up to Skolem functions and constants renaming) to the Herbrand Base of $SNF(\Gamma(C_i^\top))$.

As we saw before the concept *dummy* is unsatisfiable. We noticed informally that its unsatisfiability can only be proven by taking into account the properties of the friends of the friends of a: All the members of this class should be doctors, while one of them is not a doctor. As a consequence, *dummy* is unsatisfiable, but its simplified versions $dummy_0^\top$ and $dummy_1^\top$ are both satisfiable. As a semantic counterpart, we notice that $SNF(\Gamma(dummy))$ is unsatisfiable and this follows from the existence of a term of the form $f(c)$ which denotes the friend of the friend c of a. Actually $SNF(\Gamma(dummy))$ is both S_0-3- and S_1-3-satisfiable, where a concept C is S_i-3-satisfiable if $SNF(\Gamma(C))$ is S-3-satisfiable with $S = S_i$. This can be easily shown by noticing that the atom $D(f(c))$ does not belong to S_1, hence it is possible to define an S_1-3-interpretation $I = \langle I^+, I^- \rangle$ such that $D(f(c)) \in I^+$ and $D(f(c)) \in I^-$. This kind of interpretation "hides" the reason of inconsistency of the concept *dummy*, which is therefore S_1-3-satisfiable. This argument obviously holds also for the stratum S_0.

The above example shows that it is possible to relate S_i-3-satisfiability of a concept C of $\mathcal{ALE}$ to satisfiability of the ith element $C_i^\top$ of the sequence $\sigma^\top$. The following theorem formalizes this result.

Theorem 7.5 (Semantics of $\sigma^\top$). *For all i $(0 \leqslant i \leqslant n+1)$, C is S_i-3-satisfiable iff $C_i^\top$ is satisfiable.*

What this theorem says is that our approximation schema based on the analysis of syntactically simplified concepts readily corresponds to the semantic idea of focusing on subsets of the Herbrand Base which are defined by a simpler universe. This characterization of the approximation process could not be obtained if we were to use the more

standard semantics for $\mathcal{AL}$-languages (see Appendix A.4) based on extension functions, since we could not introduce the notion of complexity of terms.

The above correspondence between S_i-3-satisfiability of a concept C of $\mathcal{ALE}$ and satisfiability of the ith element $C_i^{\top}$ of the sequence $\sigma^{\top}$ can be easily extended to the dual case of S_i-1-satisfiability. Similarly to the previous case, it is possible to show that the approximation of the satisfiability of a concept C through the sequence $\sigma^{\perp}$ can be interpreted as S_i-1-satisfiability of the translated concept $SNF(\Gamma(C))$. This follows from the equivalence of the Herbrand Base of $SNF(\Gamma(C_i^{\perp}))$ and the Herbrand Base of $SNF(\Gamma(C_i^{\top}))$.

Theorem 7.6 (Semantics of $\sigma^{\perp}$). *For all i $(0 \leqslant i \leqslant n+1)$, C is S_i-1-satisfiable iff $C_i^{\perp}$ is satisfiable.*

Notice that the difference between S_i-1- and S_i-3-interpretations relies on the interpretation of atoms not belonging to the stratum S_i. As an example, in each S_1-1-interpretation I of the concept *dummy* it holds that $D(f(c)) \notin I^+$ and $D(f(c)) \notin I^-$.

We conclude this section by making some considerations about the choice of the subsets S_i of the Herbrand Base of $SNF(\Gamma(C_i^{\top}))$ defining our approximation schema. Donini et al. show in [25] that the number of primitive concepts and roles used is not a source of complexity in $\mathcal{ALE}$. More precisely, deciding the satisfiability of an $\mathcal{ALE}$ concept C is a co-NP-complete problem even if a single primitive role and no primitive concepts but $\top$ occur in C.

This fact has an important impact in the choice of an approximation schema for $\mathcal{ALE}$ concepts. In particular it shows that, if the subset S of the Herbrand Base contains all the ground instances of a single atomic role, then deciding S-3-satisfiability of an $\mathcal{ALE}$ concept is still co-NP-complete. Therefore the sets S defining the approximation schema must be designed so that the real source of complexity—which is the depth of concepts—is addressed.

7.4. Approximation in $\mathcal{ALC}$

In this subsection we define a method for approximating the task of deciding satisfiability of an $\mathcal{ALC}$ concept. Like the method illustrated in the previous subsection, also this one is based on a syntactic manipulation of concepts that is interpreted in terms of S-1- and S-3-satisfiability.

A first obvious question is: can we plainly use the method introduced in Section 7.3 for the purpose of approximating $\mathcal{ALC}$ concepts? We notice that $\mathcal{ALC}$ without existential quantifiers has at least the expressiveness of propositional calculus. As a consequence deciding satisfiability of $\mathcal{ALC}$ concepts without existential quantifiers is an NP-hard problem. This implies that the method defined for the approximation of an $\mathcal{ALE}$ concept leads—when used for $\mathcal{ALC}$ concepts—to polynomially intractable problems from the very first step of the approximation. In our opinion an effective method for the approximation of $\mathcal{ALC}$ should address both sources of complexity of this language (see [26]): the complexity of existentials, also present in $\mathcal{ALE}$, and that of disjunction, present in the existential-free fragment of $\mathcal{ALC}$.

The source of complexity of disjunction is also present in the propositional calculus. The method presented in Section 4 for approximating propositional satisfiability relies on the use of S-1- and S-3-satisfiability, in which sets S containing more and more boolean variables are taken into account. This corresponds to a stratification $S_0 \subseteq S_1 \subseteq \cdots \subseteq S_{n+1} = B_h$ of the Herbrand Base B_h, in which each stratum contains all the ground instances of a predicate symbol. As we noticed at the end of the previous section, such a stratification cannot be used for the approximation of $\mathcal{ALE}$, which must instead be based on the complexity of the terms of the Herbrand Universe. It is therefore natural to combine both ideas for the approximation of $\mathcal{ALC}$. In the method we are going to present, the concepts of the approximating sequences have both a fixed depth and interpret classically only a subset of the primitive concepts.

Let's start again with a concrete example and consider the following $\mathcal{ALC}$ concept:

$$D \equiv (A \sqcap \neg A) \sqcup ((\forall R.A) \sqcap \exists R.\neg A).$$

D is unsatisfiable, because it is the union of two unsatisfiable $\mathcal{ALE}$ concepts. D can be approximated either by substituting the existentially quantified subconcept $\exists R.\neg A$ with $\top$ or by replacing the primitive concept A and its negation $\neg A$ with $\top$. Moreover both methods can be combined. Each of the following $\mathcal{ALC}$ concepts is weaker than D and satisfiable:

$$D^{\top}_{\emptyset,0} \equiv (\top \sqcap \top) \sqcup (\forall R.\top \sqcap \top).$$
$$D^{\top}_{\{A\},0} \equiv (A \sqcap \neg A) \sqcup (\forall R.A \sqcap \top).$$
$$D^{\top}_{\emptyset,1} \equiv (\top \sqcap \top) \sqcup (\forall R.\top \sqcap \exists R.\top).$$

The first subscript denotes the set of primitive concepts that are not substituted by $\top$, while the second one denotes the depth of the approximation. In general, given an $\mathcal{ALC}$ concept C of depth n built on the set $\mathcal{A}$ of primitive concepts, a set $P \subseteq \mathcal{A}$ and an index $0 \leqslant i \leqslant n+1$, we denote with the symbol $C^{\top}_{P,i}$ the $\mathcal{ALC}$ concept obtained from C by means of the following rules:

(1) Substitute each (positive or negative) occurrence of a primitive concept not in P with the concept $\top$, thus obtaining the concept C'.
(2) Substitute every existentially quantified subconcept of C' which is in the scope of at least i universal quantifiers with the concept $\top$.

The concept $C^{\bot}_{P,i}$ is obtained by substituting $\bot$ instead of $\top$. Let P_1 and P_2 be two subsets of $\mathcal{A}$ and let i_1 and i_2 be two indexes such that $0 \leqslant i_1, i_2 \leqslant n+1$. Let C be an $\mathcal{ALC}$ concept and let $C_1 \equiv C^{\top}_{P_1,i_1}$ and $C_2 \equiv C^{\top}_{P_2,i_2}$ be two approximations of C. We say that $C_1 \preceq C_2$ holds iff both $P_1 \subseteq P_2$ and $i_1 \leqslant i_2$ hold. We define the relation $\preceq$ only between pairs of weaker approximations and pairs of stronger ones. This relation is a partial order in the set of the approximations of an $\mathcal{ALC}$ concept. The following are two interesting properties of this relation:

Theorem 7.7 (Monotonicity). *Let P be a subset of $\mathcal{A}$ and i be an index such that $0 \leqslant i \leqslant n+1$. If $C^{\top}_{P,i}$ is unsatisfiable, then any subconcept D of C such that $C^{\top}_{P,i} \preceq D$ is unsatisfiable. If $C^{\bot}_{P,i}$ is satisfiable, then any subconcept D of C such that $C^{\bot}_{P,i} \preceq D$ is satisfiable.*

Observation 7.8 (Convergence). *If C is unsatisfiable, then there exists a subset P of $\mathcal{A}$ and an index i $(0 \leqslant i \leqslant n+1)$ such that $C^{\top}_{P,i}$ is unsatisfiable. If C is satisfiable, then there exists a subset P of $\mathcal{A}$ and an index i $(0 \leqslant i \leqslant n+1)$ such that $C^{\bot}_{P,i}$ is satisfiable.*

The above properties are analogous to Theorem 7.3 and Observation 7.4 for $\mathcal{ALE}$. Along the same lines of Section 7.3, we say that any $\mathcal{ALC}$ concept C can be approximated by means of an increasing (with respect to $\preceq$) sequence of weaker concepts $C^{\top}_{P,i}$. This corresponds to a weak approximation of C. A strong approximation can be obtained by considering an increasing sequence of concepts $C^{\bot}_{P,i}$.

It can be shown that satisfiability checking of a subconcept $C^{\top}_{P,i}$ or $C^{\bot}_{P,i}$ can be done in time proportional to $2^{|P|\cdot i}$. This property entitles us to perform an argument similar to that of Section 7.3, in order to show that our method for checking satisfiability of $\mathcal{ALC}$ concepts in an incremental fashion has a complexity comparable to the standard algorithms (see [68]).

From the semantical point of view it is possible to characterize this form of approximation in terms of S-3- and S-1-interpretations. Let C be an $\mathcal{ALC}$ concept and n be its depth—defined exactly as in the case of $\mathcal{ALE}$ concepts. The stratification $U_0 \subseteq U_1 \subseteq \cdots \subseteq U_{n+1} = U_h$ of the Herbrand Universe U_h of $SNF(\Gamma(C))$ is defined as in the previous section. We define the subset $S_{P,i}$ of the Herbrand Base B_h of $SNF(\Gamma(C))$ to be the following set:

$$\{A(t) \mid A \text{ is a primitive concept in } P \text{ and } t \in U_i\}$$
$$\cup \{R(t_1, t_2) \mid R \text{ is a role and } t_1, t_2 \in U_i\}.$$

We say that C is $S_{P,i}$-3-satisfiable iff it is S-3-satisfiable for $S = S_{P,i}$. The intuition behind this definition is that we are confining our attention only to a subset of the set S_i defined in Section 7.3. In particular we want a unary atom $A(t)$ to be in $S_{P,i}$ only if the concept A belongs to the set of privileged primitive concepts P. In this way we are limiting both sources of potential complexity: disjunction and existential quantification. Contradiction cannot arise from roles, since $\mathcal{ALC}$ does not support negation on roles. In an analogous way we define $S_{P,i}$-1-satisfiability. The following properties formalize the relations between syntax and semantics:

Theorem 7.9 (Semantics of $\top$ rewriting). *For all i $(0 \leqslant i \leqslant n+1)$ and $P \subseteq \mathcal{A}$ the concept C is $S_{P,i}$-3-satisfiable iff $C^{\top}_{P,i}$ is satisfiable.*

Theorem 7.10 (Semantics of $\bot$ rewriting). *For all i $(0 \leqslant i \leqslant n+1)$ and $P \subseteq \mathcal{A}$ the concept C is $S_{P,i}$-1-satisfiable iff $C^{\bot}_{P,i}$ is satisfiable.*

We conclude this section by noticing that the method we propose for the approximation of $\mathcal{ALC}$ concepts is in some sense underspecified, since we gave no criteria for choosing subsets P of the set of primitive concepts. In particular, it makes sense to ask whether it is more appropriate to enlarge the set P or to increase the index i when deciding satisfiability of a given concept. We remind that for each $A \in P$ and each term $t \in U_i$ we know exactly whether t satisfies the property A or not.

In Section 7.3 we noticed that increasing the index i corresponds to admit terms of the interpretation that are more and more complex, i.e. to have a wider Herbrand Universe. On the other hand enlarging the set P amounts to interpret classically a wider set of concepts, but we don't have to consider new terms, i.e. the Herbrand Universe remains the same. Therefore if we want a sound and complete treatment of a large collection of individuals, but we are not committed to assign them many properties, then we should increase the index i. If we want a sound and complete treatment of a large collection of properties, but we are not committed to consider all the potential individuals, then we should enlarge the set P.

All the results presented in this section will be proved in Appendix A.4.

The results of this section have been published in a preliminary form in [13].

8. Propositional modal logics

In this section we present the use of our approximation methods in the field of modal logics. In particular we show how it is possible to approximate satisfiability in the most widely used modal logics for knowledge and belief, namely $\mathcal{S}5$, $\mathcal{K}$, $\mathcal{T}$ and $\mathcal{S}4$. This is the second section devoted to modal logics. In Section 6 we focused on the use of a multimodal language as a tool for modeling approximate knowledge of a resource-bounded agent. The goal of this section is to approximate reasoning in *classical* modal logics.

A detailed analysis of the computational complexity of satisfiability problems in several propositional modal systems has been done by Ladner [52]. He showed that the problem of checking satisfiability of a formula in the systems $\mathcal{K}$, $\mathcal{T}$ and $\mathcal{S}4$ is PSPACE-complete, while the same problem is NP-complete in the system $\mathcal{S}5$. Therefore most of the modal logics frequently used for modeling knowledge and belief (see [43]) lead to computationally intractable reasoning problems.

In this section we focus on the problem of applying approximation techniques to such propositional modal systems. The main idea is to extend the method defined for propositional logic by defining two classes of interpretations which are approximations of the standard Kripke semantics.

In the following we refer to modal formulae which are built on the set L^* by means of the usual connectives $\vee$ and $\wedge$, the modal operator $\boldsymbol{K}$, the negation $\neg\boldsymbol{K}$ of the modal operator, plus parentheses. Using the terminology introduced in Section 6, we call formulae of this kind modal negation normal form (MNNF) formulae. Let α be a formula in which each occurrence of a modal operator lies in the scope of at most n modal operators. The parameter n is called the modal depth of α.

We remind the definition of a Kripke model.

Definition 8.1 (*Kripke* [51]). A Kripke model is a triple $\mathcal{M} = \langle W, R, V\rangle$ where W is a set of worlds, R an accessibility relation among worlds and V a mapping $W \rightarrow \tau$, where τ is the set of all the truth assignments of L^*.

In a standard Kripke model $V(w)$ is a 2-interpretation for every world $w \in W$. We

refer to standard Kripke models as 2-*Kripke interpretations*. In this section we want to consider also other forms of interpretations, defined as follows.

Definition 8.2 (*S*-1-*Kripke interpretation*). An *S*-1-*Kripke interpretation* is a Kripke model such that $V(w)$ is an *S*-1-interpretation for every world $w \in W$ (cf. Definition 4.3).

Definition 8.3 (*S*-3-*Kripke interpretation*). An *S*-3-*Kripke interpretation* is a Kripke model such that $V(w)$ is an *S*-3-interpretation for every world $w \in W$ (cf. Definition 4.1).

The evaluation of a propositional formula γ in any world $w \in W$ of an *S*-1-Kripke interpretation $\mathcal{M} = \langle W, R, V \rangle$ is defined as in Section 6; in particular we write $\mathcal{M}, w \models^1_S \gamma$ iff $V(w)(\gamma) = 1$, that is $V(w)$ maps γ into 1. The value assigned by $\mathcal{M}$ to a MNNF formula α in a world $w \in W$ is defined by using the rule for propositional formulae and recursively the following rules:

- $\mathcal{M}, w \models^1_S \boldsymbol{K}\beta$ iff $\forall t \in W\ wRt$ implies $\mathcal{M}, t \models^1_S \beta$;
- $\mathcal{M}, w \models^1_S \neg\boldsymbol{K}\beta$ iff $\exists t \in W\ wRt$ and $\mathcal{M}, t \models^1_S \neg\beta$.

Notice that β may not be in MNNF, in this case we need to transform it in its MNNF equivalent. A modal formula α is *S*-1-*Kripke satisfiable* iff there exists an *S*-1-Kripke interpretation $\mathcal{M} = \langle W, R, V \rangle$ and a $w \in W$ s.t. $\mathcal{M}, w \models^1_S \alpha$.

The definitions of *S*-3-Kripke interpretation and *S*-3-Kripke satisfiability are straightforward. We refer to standard satisfiability of a modal formula as 2-*Kripke satisfiability*.

Some remarks are in order here to make clear the differences and the analogies with the system introduced in Section 6. Notice, first of all, that the language introduced in Section 6 is an extension of the classical modal language with one modal operator, in fact we have two families of modal operators, i.e. $\Box^3_S$ and $\Box^1_S$, which are used as meta-descriptors of the entailment relations $\models^3_S$ and $\models^1_S$. Here, on the other hand, we are dealing with a simple modal language. Furthermore, here our main concern is the computational behavior of the problems of satisfiability testing, while in Section 6 we were mainly interested in the expressiveness of the language as a representational tool for approximate inference.

This difference in the emphasis leads to two different semantic definitions for negation in front of modal operators. To be more concrete, let α be a propositional formula, it is always the case that $\Box^3_S\alpha \wedge \neg\, \Box^3_S\, \alpha$ is unsatisfiable while $\boldsymbol{K}\alpha \wedge \neg\boldsymbol{K}\alpha$ is *S*-3-Kripke satisfiable whenever $\alpha \wedge \neg\alpha$ is *S*-3-satisfiable. This is an immediate consequence of the two different definitions for negation in fact here $\mathcal{M}, w \models^3_S \neg\boldsymbol{K}\beta$ holds iff we can find a world connected to w which makes β false, while $\mathcal{M}, w \models \neg\, \Box^3_S\, \alpha$ holds iff we can find an *S*-3-situation connected to w which does not make α true, i.e. $\mathcal{M}, w \not\models \Box^3_S\alpha$.

We want to point out that, if we were to choose the second definition for *S*-3-Kripke satisfiability then the results proven in the sequel on uniform complexity would no longer hold. On the other side, if we were to define satisfiability of negated modal formulae in Section 6 using the rule used here, the resulting language would not be expressive enough to be able to represent the approximate entailment relations.

We now demonstrate our definitions by means of two examples.

Example 8.4 (*Proving S-3-Kripke unsatisfiability*). We show an alphabet L, a modal formula σ on L and a subset S of L such that σ is S-3-Kripke unsatisfiable.

Let L be $\{a, b, c\}$, S be $\{a, b\}$ and σ be $(\boldsymbol{K}a \wedge \boldsymbol{K}(\neg a \vee b) \wedge \boldsymbol{K}\neg b \wedge \neg \boldsymbol{K}c)$. Let us assume that σ is S-3-Kripke satisfiable. By the above definition this implies that there exists an S-3-Kripke-interpretation $\mathcal{M} = \langle W, R, V\rangle$ and a $w \in W$ such that all the following conditions hold:

- $\mathcal{M}, w \models^3_S \boldsymbol{K}a$;
- $\mathcal{M}, w \models^3_S \boldsymbol{K}(\neg a \vee b)$;
- $\mathcal{M}, w \models^3_S \boldsymbol{K}\neg b$;
- $\mathcal{M}, w \models^3_S \neg \boldsymbol{K}c$.

The above conditions are equivalent to the following ones:

- $\forall t \in W\ wRt$ implies $\mathcal{M}, t \models^3_S a$;
- $\forall t \in W\ wRt$ implies $\mathcal{M}, t \models^3_S (\neg a \vee b)$;
- $\forall t \in W\ wRt$ implies $\mathcal{M}, t \models^3_S \neg b$;
- $\exists t \in W\ wRt$ and $\mathcal{M}, t \models^3_S \neg c$.

Let t_0 be the world whose existence is implied by the last condition, that is let $t_0 \in W$, wRt_0 and $V(t_0)(\neg c) = 1$. According to the other conditions, the truth assignment $V(t_0)$ must satisfy the propositional formula $(a \wedge (\neg a \vee b) \wedge \neg b)$. Taking into account that $V(t_0)$ is an S-3-interpretation of L^* and that $S = \{a, b\}$, $V(t_0)$ satisfies $(a \wedge \neg b)$ if and only if it maps a into 1, $\neg a$ into 0, b into 0 and $\neg b$ into 1. Therefore $V(t_0)$ maps $(\neg a \vee b)$ into 0, hence it does not satisfy $(a \wedge (\neg a \vee b) \wedge \neg b)$. This contradiction proves that σ is S-3-Kripke unsatisfiable.

Notice that σ is S'-3-Kripke satisfiable, where $S' = \{a\}$.

Example 8.5 (*Proving S-1-Kripke satisfiability*). We show an alphabet L, a modal formula τ on L and a subset S of L such that τ is S-1-Kripke satisfiable.

Let L be $\{a, b\}$, S be $\{b\}$ and τ be $(\neg b \wedge \boldsymbol{K}a \wedge \boldsymbol{K}(\neg a \vee b))$. By the above definition, τ is S-1-Kripke satisfiable if and only if there exists an S-1-Kripke-interpretation $\mathcal{M} = \langle W, R, V\rangle$ and a $w \in W$ such that all the following conditions hold:

- $\mathcal{M}, w \models^1_S \neg b$;
- $\forall t \in W\ wRt$ implies $\mathcal{M}, t \models^1_S a$;
- $\forall t \in W\ wRt$ implies $\mathcal{M}, t \models^1_S (\neg a \vee b)$.

Let W be the singleton $\{w\}$, R be the empty set and the S-1-interpretation $V(w)$ of L^* be such that $V(w)(a) = V(w)(\neg a) = V(w)(b) = 0$ and $V(w)(\neg b) = 1$. It is easy to see that $\mathcal{M}, w \models^1_S \tau$ holds, where $\mathcal{M} = \langle W, R, V\rangle$. Therefore τ is S-1-Kripke satisfiable.

We now show two straightforward consequences of the definitions of S-1 and S-3-Kripke satisfiability (α is a MNNF formula).

Theorem 8.6 (Monotonicity). *For any S and S' such that $S \subseteq S' \subseteq L$, if α is S-1-Kripke satisfiable, then α is S'-1-Kripke satisfiable (hence 2-Kripke satisfiable). Moreover if α is S-3-Kripke unsatisfiable, then α is S'-3-Kripke unsatisfiable (hence 2-Kripke unsatisfiable).*

Observation 8.7 (Convergence). *If α is 2-Kripke satisfiable, then there exists an $S \subseteq L$ such that α is S-1-Kripke satisfiable. If α is 2-Kripke unsatisfiable, then there exists an $S \subseteq L$ such that α is S-3-Kripke unsatisfiable.*

The above properties account for a stepwise procedure for deciding 2-Kripke satisfiability of a modal formula, which is analogous to that defined in Section 4 for checking 2-satisfiability of a propositional formula. As a consequence of Theorem 8.6 we have that the formula σ of Example 8.4 is 2-Kripke unsatisfiable and the formula τ of Example 8.5 is 2-Kripke satisfiable. The following theorem shows that there exist modal systems, e.g. $\mathcal{S}5$, in which such a stepwise procedure is interesting from a computational point of view.

Theorem 8.8 (Uniform complexity for $\mathcal{S}5$). *If we restrict our attention to accessibility relations which are reflexive, transitive and euclidean, then there exists one algorithm to decide if α is S-1-Kripke satisfiable and one to decide if α is S-3-Kripke satisfiable both running in $O(m \cdot |\alpha| \cdot 2^{|S|})$ time, where m is the number of occurrences of the modal operator in α.*

The above theorem shows that all the considerations made in Section 6 on the approximation of the 2-satisfiability of a propositional formula also hold for the approximation of the 2-Kripke satisfiability of any formula of the modal system $\mathcal{S}5$.

The same idea can be applied, with only minor variations, to other systems whose satisfiability check is known to be an NP-complete problem, such as $\mathcal{K}45$ and $\mathcal{KD}45$. This holds since in these systems any satisfiable formula is satisfied by a 2-Kripke interpretation whose set of worlds W has size bounded by a polynomial function of the size of the formula itself.

On the other hand, as proved by the following result, there exist interesting modal systems, such as $\mathcal{K}$, in which the stepwise procedure suggested by Theorem 8.6 and Observation 8.7 is not useful from a computational point of view.

Theorem 8.9 (Non-uniform complexity for $\mathcal{K}$). *If the accessibility relation is unrestricted, then deciding if α is S-1-Kripke satisfiable and deciding if α is S-3-Kripke satisfiable are PSPACE-complete problems even if $|S| = 1$.*

This result prevents us from the development of a result analogous to Theorem 8.8 for unrestricted accessibility relation (unless P=PSPACE). This is not surprising, since Ladner has shown [52] that there exist formulae in the systems $\mathcal{K}$, $\mathcal{T}$ and $\mathcal{S}4$, which are satisfied only by 2-Kripke interpretations having a set of worlds whose size is exponential in the nesting of the modal operators.

A possible way to overcome this problem is to focus only on limited parts of the interpretations. We now present a semantics for approximation which further extends the possible-worlds semantics. The idea is that a Kripke interpretation $\mathcal{M}$ should satisfy a formula α in a world w iff α is satisfied in the subset $W' \subseteq W$ of the possible worlds containing only those worlds whose *distance* from w is less than or equal to i, where i is a particular integer. In this way we can limit our attention to Kripke interpretations

having $O(2^i)$ worlds. The worlds which are outside the "range" i are treated differently in the S-1- and in the S-3-case. In particular S-1-Kripke interpretations are "pessimistic", since they do not validate anything in those worlds, while S-3-Kripke interpretations are "optimistic", since they validate everything.

Let α be a modal formula, n its modal depth, S a subset of L, $i \leqslant n+1$, and $\mathcal{M} = \langle W, R, V\rangle$ an S-3-Kripke interpretation. We define a new relation $\models^3_{S,i}$ as follows (γ is a propositional formula):

- $\mathcal{M}, w \models^3_{S,i} \gamma$ if and only if $(V(w)(\gamma) = 1$ or $i < 0)$;
- $\mathcal{M}, w \models^3_{S,i} \alpha \wedge \beta$ if and only if $\mathcal{M}, w \models^3_{S,i} \alpha$ and $\mathcal{M}, w \models^3_{S,i} \beta$;
- $\mathcal{M}, w \models^3_{S,i} \alpha \vee \beta$ if and only if $\mathcal{M}, w \models^3_{S,i} \alpha$ or $\mathcal{M}, w \models^3_{S,i} \beta$;
- $\mathcal{M}, w \models^3_{S,i} \boldsymbol{K}\alpha$ if and only if $\forall t \in W\ wRt$ implies $\mathcal{M}, t \models^3_{S,i-1} \alpha$;
- $\mathcal{M}, w \models^3_{S,i} \neg\boldsymbol{K}\alpha$ if and only if $\exists t \in W\ wRt$ and $\mathcal{M}, t \models^3_{S,i-1} \neg\alpha$.

Notice that according to the above definition, if $i < 0$ then any formula is true in any world. A modal formula α is $\langle S, i\rangle$-3-Kripke satisfiable iff there exists an S-3-interpretation $\mathcal{M} = \langle W, R, V\rangle$ and a $w \in W$ s.t. $\mathcal{M}, w \models^3_{S,i} \alpha$.

A similar definition can be given for the relation $\models^1_{S,i}$. The only difference is that now $\mathcal{M}$ is an S-1-Kripke interpretation and the definition of the base case is:

- $\mathcal{M}, w \models^1_{S,i} \gamma$ if and only if $(V(w)(\gamma) = 1$ and $i \geqslant 0)$.

Notice that if $i < 0$ then a formula cannot be true in a world. It is easy to show that the analogue of Theorem 8.6 holds for the new definitions, when we compare pairs $\langle S, i\rangle$ and $\langle S', j\rangle$ such that $S \subseteq S' \subseteq L$ and $i \leqslant j \leqslant n+1$. In other words if a modal formula α is $\langle S, i\rangle$-3-Kripke unsatisfiable, then it is $\langle S', j\rangle$-3-Kripke unsatisfiable, and if it is $\langle S, i\rangle$-1-Kripke satisfiable, then it is $\langle S', j\rangle$-1-Kripke satisfiable. Moreover, there exists a subset S of L and an integer $i \leqslant n+1$ such that α is either $\langle S, i\rangle$-1-satisfiable or $\langle S, i\rangle$-3-unsatisfiable.

We can also prove that there exists an algorithm for deciding if a modal formula α is $\langle S, i\rangle$-3-Kripke satisfiable which runs in $O(|\alpha| \cdot 2^{|S|\cdot i})$ time, provided that the constraints of either $\mathcal{K}$ or $\mathcal{T}$ or $\mathcal{S}4$ hold on the accessibility relation. The algorithm for determining $\langle S, i\rangle$-3-Kripke satisfiability of a modal formula α is based on a mapping of α into another modal formula $\psi^3_{S,i}(\alpha)$, in which the nesting of the modal operators is limited and any occurrence of a letter not in S is substituted by the literal t. More precisely, if $i < 0$ then $\psi^3_{S,i}(\alpha)$ is t, otherwise $\psi^3_{S,i}(\alpha)$ is obtained by:

(1) substituting each occurrence of a letter in $L \setminus S$ with the literal t, thus obtaining the formula α';
(2) substituting every subformula $\neg\boldsymbol{K}\beta$ of α' which is in the scope of at least i modal operators $\boldsymbol{K}$ with the literal t.

The relation between α and $\psi^3_{S,i}(\alpha)$ is the following:

Theorem 8.10 (Semantics of rewriting). *Let α be a modal formula and X be a modal system admitting the K schema and the rule of necessitation. The formula α is $\langle S, i\rangle$-3-Kripke satisfiable in the system X iff $\psi^3_{S,i}(\alpha)$ is 2-Kripke satisfiable in the system X.*

Since the 2-Kripke satisfiability of $\psi^3_{S,i}(\alpha)$ can be determined with standard algorithms —which run in time $O(|\alpha| \cdot 2^{|S| \cdot i})$—we have an effective procedure to decide $\langle S, i\rangle$-3-Kripke satisfiability. The algorithm for checking $\langle S, i\rangle$-1-Kripke satisfiability is based on a similar mapping $\psi^1_{S,i}$ in which the literal f is used instead of using t.

Theorem 8.10 allows us to extend all the considerations on the approximation of the 2-satisfiability of a propositional formula to the approximation of the 2-Kripke satisfiability of any formula of the modal systems $\mathcal{K}$, $\mathcal{T}$ and $\mathcal{S}4$.

As for the strategy for approximating satisfiability of a formula in the modal systems $\mathcal{K}$, $\mathcal{T}$ or $\mathcal{S}4$, we have to choose whether to increase the index i or to enlarge the set S. Given that the index i captures the complexity of the frame, while the set S captures the accuracy of each world within the frame, the discussion presented at the end of Section 7.4 applies to this case as well.

All the results presented in this section will be proved in Appendix A.5.

The results of this section have been published in a preliminary form in [12].

9. Conclusions, open problems and future research

Approximation techniques are widely used in many areas of computer science to deal with polynomially intractable problems. In this paper we dealt with approximations in the non-numerical problems of inference and satisfiability checking in a number of logical systems.

We started by analyzing two techniques for approximate reasoning that have been defined in the literature by other authors. By comparing the fundamental aspects of these two techniques we were able to give a list of desiderata for a new approximation method. The most important aspects of the new method are: semantical description and efficient computation of approximate answers, possibility of having more and more precise answers, unique approach for a variety of deduction problems.

The new technique has been introduced in the framework of propositional logic, and has been successively extended to inference and satisfiability checking in other logical systems, including fragments of first-order logic and modal logic. Moreover we introduced an epistemic language for the representation of approximate knowledge, owned by an agent with limited computational resources. Further results on the applicability of our approximation schema to reasoning problems of default logic and circumscription, two of the best-known formalism for nonmonotonic reasoning, have been presented in [14].

These results complement the more classical approach to tractability, that is language restriction. In particular approximation techniques can be very useful whenever the expressiveness of polynomial languages is severely limited.

In our research we stressed the point of providing the approximate answer with a clear semantics. Other authors (for example [32, 38]) are more interested in preserving the plausibility of the mechanism as a model of human reasoning. We believe that a parameter specifying "how much" the user can trust an approximate answer would be very interesting from the practical point of view. In our research we pursued symbolic approaches and we tried to specify such a degree of belief by means of an epistemic

logic for non-omniscient agents (cf. Section 6). Anyway we don't disregard in principle methods based on numerical approaches.

There are a number of possible directions in which our research on approximate reasoning can be continued. We close our work by giving a list of topics that we consider particularly interesting for future research:

Logical aspects

- We would like to provide S-1- and S-3-entailment with a proof theory, for example with a Gentzen-style system.
- We would like to give a sort of "interpolation theorem" for S-3- and S-1-entailment (cf. [50, Section 56]) In such a way we expect to have a formalization of the gain of information that we have in the stepwise solution of the problem.

Computational aspects

- It would be interesting to study the computational complexity of optimization problems like:
  - Given a pair of propositional formulae T and γ, find the size $|S|$ of a minimal set S such that either $T \models^3_S \gamma$ or $T \not\models^1_S \gamma$ holds. Find one such set S of minimal size.
  - Given a pair of propositional formulae T and γ and the fact that both $T \not\models^3_S \gamma$ and $T \models^1_S \gamma$ for a fixed set of letters $S \neq \emptyset$, find the size $|S'|$ of a minimal set $S' \supset S$ such that either $T \models^3_{S'} \gamma$ or $T \not\models^1_{S'} \gamma$ holds. Find one such set S' of minimal size.

Integration with other approximation techniques

- We have briefly mentioned in Section 4 that our method has some similarities with the techniques used in the field of abstract interpretation. Our approach is more semantics-oriented, while abstract interpretation uses algebraic and syntactic methods. We believe that a more careful comparison of the relative advantages of the two methods can lead to further results.
- We think that the idea of knowledge compilation (see [47,49,69] and Section 2) is particularly interesting, as this is a task that can be done off-line. We plan to investigate about the possibility to integrate knowledge compilation and on-line approximate reasoning.
- Is there any relation between our definition of approximation and a definition of approximation based on numerical estimates? As an example, we know that $T \models \gamma$ iff $\mathcal{M}(T) \subseteq \mathcal{M}(\gamma)$, where $\mathcal{M}()$ denotes the models of a propositional formula. Let us define a new form of approximate entailment as follows: consider the number δ defined as the ratio $(|\mathcal{M}(T)| - |\mathcal{M}(\gamma)|)/|\mathcal{M}(T)|$. Clearly $0 \leqslant \delta \leqslant 1$ and $T \models \gamma$ iff $\delta = 0$. Intuitively we can say that an estimate of δ gives an estimate of the validity of the relation $T \models \gamma$; in particular if we suspect that δ is low, then we might be willing to accept that $T \models \gamma$ holds.
  Is there any relation between δ and S-3-, S-1-entailment? Notice that if we can relate a numerical parameter of this kind with the approximate inference, then we might have a heuristic "measure" of the reliability of intermediate answers.

Use of meta-knowledge

- In Section 6 we have presented some ideas on how an expressive language can be used to declaratively state properties of a knowledge base and help in the query answering process. In our opinion, this is a very important issue and we believe that it deserves further theoretical investigation as well as experiments in applicative domains. Moreover the computational complexity of such a modal language has to be investigated.
- Extend the epistemic language for non-omniscient agents to a multi-agent framework. Such an extended language would be able to formalize the interaction between several non-omniscient agents.

Approximation of other computational tasks

- Extend the work on approximation to planning problems (which are in general search problems). In this field, there is a need for efficient inference mechanisms to be used in real-time systems. In the literature some interesting work on approximation of planning has already been done (see for example literature on any-time algorithms [23, 39, 65]). We believe that our framework can be successfully applied to planning problems by adopting a stepwise construction of plans.

Acknowledgments

We want to thank Marta Cialdea, Francesco M. Donini, Torsten Schaub, Carolyn Talcott and an anonymous referee for their useful suggestions which helped improving both the presentation and the contents of the paper. Luigia Carlucci Aiello, Maurizio Lenzerini and Andrea Schaerf carefully read and commented previous versions of this paper. Marco Cadoli acknowledges John McCarthy and Carolyn Talcott for their hospitality at the Computer Science Department of the Stanford University, where part of this research has been developed. Furthermore, we are grateful to Bart Selman and Henry Kautz for clarifications on their method.

Appendix A. Proofs of theorems

A.1. Appendix to Section 4

First of all we prove two lemmata that originally appeared in [56] without proof. The lemmata have already been referenced in Section 2.1.

Lemma A.1 (Soundness of $\models^3$). *If $T \models^3 \gamma$ holds, then $T \models \gamma$ holds.*

Proof. Each 3-interpretation of T is also an interpretation of T. Therefore the set of interpretations satisfying T is a subset of the set of 3-interpretations satisfying T. The same holds for γ. Hence if γ is true in all the 3-interpretations satisfying T, then it is also true in all the interpretations satisfying T. $\Diamond$

Lemma A.2 (Polynomiality of $\models^3$). *$T \models^3 \gamma$ holds iff either a clause subsumed by γ (i.e. such that all its literals also occur in γ) occurs in T or a pair $p, \neg p$ of literals occurs in γ. Therefore, determining if $T \models^3 \gamma$ holds can be checked in $O(|T| \cdot |\gamma|)$ time.*

Proof. Let L' be the alphabet $L \cup \{\overline{p} \mid p \in L\}$. Let T' be the formula obtained from T by substituting each occurrence of a negative literal $\neg p$ with the corresponding letter $\overline{p}$ of L'. Let γ' be the clause obtained from γ in the same way. Let T'' be the formula obtained by conjoining T' with all the clauses of the set $\{(p \vee \overline{p}) \mid p \in L\}$. Let γ'' be the clause obtained from γ' in the same way. It is easy to notice that there is a 1–1 correspondence between the set of 3-interpretations of T and the set of interpretations of T'': in each 3-interpretation of T the unique constraint on the truth values of a pair of corresponding literals $p, \neg p$ is that they cannot be both 0; moreover, in each interpretation of T'' the unique constraint on the truth values of a pair of corresponding literals $p, \overline{p}$ is that they cannot be both 0. The same property holds for the set of 3-interpretations of γ and the set of interpretations of γ''. Therefore $T \models^3 \gamma$ holds iff $T'' \models \gamma''$ holds. Since negative literals do not occur either in T'' or in γ'', the last relation holds iff either a clause subsumed by γ' (i.e. such that all its literals also occur in γ') occurs in T' or a pair $p, \overline{p}$ of literals occurs in γ'. Therefore $T \models^3 \gamma$ holds iff either a clause subsumed by γ occurs in T or a pair $p, \neg p$ of literals occurs in γ. This can be checked in $O(|T| \cdot |\gamma|)$ time. ◊

Theorem 4.4. *For any S and S' such that $S \subseteq S' \subseteq L$, if $T \models^3_S \gamma$ holds, then $T \models^3_{S'} \gamma$ (hence $T \models \gamma$). Moreover if $T \not\models^1_S \gamma$ holds and both a letter l of $S' \setminus S$ and its negation $\neg l$ do not occur in γ, then $T \not\models^1_{S'} \gamma$ holds (hence $T \not\models \gamma$).*

Proof. As far as $\models^3_S$ is concerned, the proof is the same as that of Lemma A.1.

As for $\models^1_S$, we prove that $T \models^1_{S'} \gamma$ implies $T \models^1_S \gamma$. Suppose that $T \models^1_{S'} \gamma$ holds and that there exists an S-1-interpretation M satisfying T but not satisfying γ. We show that this leads to a contradiction.

We build an S'-1-interpretation N of the alphabet L of T in the following way:

- for each $l \in S$, N maps l into 1 iff M maps l into 1; moreover it maps $\neg l$ into the opposite value;
- for each $l \in S' \setminus S$ such that l occurs in γ, N maps l into 0 and $\neg l$ into 1;
- for each $l \in S' \setminus S$ such that $\neg l$ occurs in γ, N maps l into 1 and $\neg l$ into 0;
- for each remaining letter l, N maps l and $\neg l$ into 0.

Notice that N is an S'-1-interpretation of L, since it maps every letter l of S' and its negation $\neg l$ into opposite values and it maps each remaining literal into 0. Moreover, it satisfies T, since M maps at least one literal per clause of T into 1, and the set of literals that N maps into 1 is a superset of the set of literals M maps into 1. It is easy to notice that N does not satisfy γ, but this contradicts the hypothesis that $T \models^1_{S'} \gamma$ holds. ◊

Theorem 4.6. *There exists an algorithm for deciding if $T \models^3_S \gamma$ and deciding if $T \models^1_S \gamma$ which runs in $O(|T| \cdot |\gamma| \cdot 2^{|S|})$ time.*

Proof. As far as $\models^3_S$ is concerned, forthcoming Theorem A.3 states that $T \models^3_S \gamma$ can be tested by performing $2^{|S|}$ tests of the kind $T \models^3 \delta$, where δ has size proportional to the size of γ. Using Lemma A.2, we obtain the desired upper bound for the complexity of $\models^3_S$.

As for $\models^1_S$, Theorem 5.3 reduces $T \models^1_S \gamma$ to the problem of testing S-1-unsatisfiability of $T \wedge \delta$, where δ is a CNF formula with size proportional to the size of γ. As we already showed in Section 5, S-1-satisfiability of a formula Γ in NNF can be tested in time $O(|\Gamma| \cdot 2^{|S|})$ by: (1) substituting each positive or negative literal whose corresponding letter occurs in S with the propositional constant false; (2) running any satisfiability algorithm on the resulting formula. $\Diamond$

Theorem A.3 (From S-3-entailment to 3-entailment). *Let S be the set $\{a_1, \ldots, a_m\}$. $T \models^3_S \gamma$ iff $T \models^3 \gamma \vee [(a_1 \wedge \neg a_1) \vee \cdots \vee (a_m \wedge \neg a_m)]$ holds, or equivalently $T \models^3 \gamma \vee (c_1 \vee \cdots \vee c_m)$ holds for any combination $\{c_1, \ldots, c_m\}$, where each c_i $(1 \leqslant i \leqslant m)$ is either a_i or $\neg a_i$.*

Proof.

(*Only if part*) Suppose that $T \models^3_S \gamma$ holds and a set $\{c_1, \ldots, c_m\}$ exists where each c_i $(1 \leqslant i \leqslant m)$ is either a_i or $\neg a_i$, such that $T \not\models^3 c_1 \vee \cdots \vee c_m$ holds. We show that this leads to a contradiction.

If $T \not\models^3 c_1 \vee \cdots \vee c_m$ holds, then a 3-interpretation M satisfying T exists such that T does not satisfy $c_1 \vee \cdots \vee c_m$. Notice that M maps each literal c_i $(1 \leqslant i \leqslant m)$ into 0. This implies that it maps each literal $\neg c_i$ $(1 \leqslant i \leqslant m)$ into 1. Therefore, M is also an S-3-interpretation, but this contradicts the hypothesis that $T \models^3_S \gamma$.

(*If part*) Suppose that both $T \models^3 (a_1 \wedge \neg a_1) \vee \cdots \vee (a_m \wedge \neg a_m)$ and $T \not\models^3_S \gamma$. We show that this leads to a contradiction.

If $T \not\models^3_S \gamma$, then an S-3-interpretation M satisfying T and not satisfying γ exists. We build a set H of literals in the following way: for each i $(1 \leqslant i \leqslant m)$, if M maps a_i into 0, then put a_i in H; if M maps $\neg a_i$ into 0, then put $\neg a_i$ in H. Notice that exactly one literal in $\{a_i, \neg a_i\}$ $(1 \leqslant i \leqslant m)$ occurs in H. Let H be $\{c_1, \ldots, c_m\}$, where each c_i $(1 \leqslant i \leqslant m)$ is either a_i or $\neg a_i$. Since M is a 3-interpretation satisfying T, $T \not\models^3 c_1 \vee \cdots \vee c_m$ holds, but this contradicts the hypothesis that $T \models^3 (a_1 \wedge \neg a_1) \vee \cdots \vee (a_m \wedge \neg a_m)$ holds. $\Diamond$

A.2. Appendix to Section 5

Lemma 5.2. *Suppose letters(γ) $\not\subseteq S$ holds. Let S' be the set $S \cup$ letters(γ). $T \models^3_S \gamma$ holds iff $T \models^3_{S'} \gamma$ holds.*

Proof.

(*Only if part*) See Theorem 4.4.

(*If part*) Suppose both $T \not\models^3_S \gamma$ and $T \models^3_{S'} \gamma$. We will show that this leads to a contradiction. An S-3-interpretation M exists such that M satisfies T and it does not satisfy γ. It follows that M maps each literal of γ into 0. This implies that M maps

each literal of $\neg\gamma$ into 1, therefore M is also an S'-3-interpretation, but this contradicts the hypothesis that $T \models^3_{S'} \gamma$ holds. $\Diamond$.

Theorem 5.3. *Let γ be $\gamma_S \vee \gamma_{\overline{S}}$, where both letters$(\gamma_S) \subseteq S$ and letters$(\gamma_{\overline{S}}) \cap S = \emptyset$ hold. $T \models^1_S \gamma$ holds iff $T \cup \{\neg\gamma_S\}$ is not S-1-satisfiable.*

Proof.

(*Only if part*) Suppose that $T \cup \{\neg\gamma_S\}$ is S-1-satisfiable; let M be the S-1-interpretation satisfying it. M satisfies T, but it does not satisfy γ_S. Moreover, M does not satisfy $\gamma_{\overline{S}}$ because it maps all its literals into 0. Therefore $T \not\models^1_S \gamma_S \vee \gamma_{\overline{S}}$ holds.

(*If part*) Suppose that $T \cup \{\neg\gamma_S\}$ is not S-1-satisfiable and that $T \not\models^1_S \gamma$ holds. We show that this leads to a contradiction. An S-1-interpretation M exists such that M satisfies T and does not satisfy γ. M maps each literal of γ into 0. Since no literal of γ_S is in $L \setminus S$, M satisfies $\neg\gamma_S$. Therefore M satisfies $T \cup \{\neg\gamma_S\}$, but this contradicts the former hypothesis. $\Diamond$

Theorem 5.4. *Let letters$(\gamma) \subseteq S$ hold. $T \models^3_S \gamma$ holds iff $T \cup \{\neg\gamma\}$ is not S-3-satisfiable.*

Proof. Analogous to that of Theorem 5.3 with $\gamma = \gamma_S$. $\Diamond$

Theorem 5.5. *Let δ be a set of literals such that letters$(\delta) \subseteq S$ holds and both a letter l and its negation $\neg l$ do not occur in δ; simplify(T, δ) is S-1-satisfiable iff $T \cup \delta$ is S-1-satisfiable. In addition, simplify(T, δ) is S-3-satisfiable iff $T \cup \delta$ is S-3-satisfiable.*

Proof.

(*If part for both properties*) Suppose that $T \cup \delta$ is S-x-satisfiable ($x = 1$ or 3). Let M be an S-x-interpretation satisfying it. M maps each literal of δ into 1. Since $\delta \subseteq S$ holds, M maps the negation of each literal of δ into 0. Therefore, given any clause β of T in which the negation of a literal of δ occurs, M satisfies one of the remaining literals of β. Taking into account that the only clauses of *simplify*(T, δ) are subclauses of those in which the negation of a literal of δ occurs, we can easily notice that M satisfies *simplify*(T, δ).

(*Only if part for both properties*) Suppose that *simplify*(T, δ) is S-x-satisfiable. Let M be an S-x-interpretation satisfying it. Notice that M satisfies every clause of T in which no literal of δ occurs. Recall that no literal of δ occurs in *simplify*(T, δ). We build an S-x-interpretation N according to the following rule: N maps any literal l of δ into 1 and its negation $\neg l$ into 0; moreover, N is equivalent to M for any remaining literal. M satisfies every clause of T in which no literal of δ occurs; therefore N satisfies the same clauses. Moreover, N satisfies all the literals in δ and every clause of T in which at least one literal of δ occurs. Therefore N satisfies $T \cup \delta$. $\Diamond$

Theorem 5.6. *A formula T is S-3-satisfiable iff T^m is S-3-satisfiable iff $\square \notin T^m$. A formula T is S-1-satisfiable iff T^m is S-1-satisfiable iff $T^m = \Omega$.*

Proof. We divide the proof into two parts, first of all we prove, at the same time, the

two equivalences: T is S-3-satisfiable iff T^m is S-3-satisfiable and T is S-1-satisfiable iff T^m is S-1-satisfiable. In the last part we prove the other two equivalences: T^m is S-3-satisfiable iff $\square \notin T^m$ and T^m is S-1-satisfiable iff $T^m = \Omega$. We prove the first part by induction on m. The base case is trivial since $T^0 = T$, in the inductive case we have:

(*If part for both properties*) Suppose that there is an S-x-interpretation I satisfying T^i $(0 < i < m)$ then I maps into 1 at least one literal of any clause of T^i. Let us assume that there is one clause β of T^{i+1} which is not satisfied by I we show that this leads to a contradiction. This clause β cannot belong to T^i because it would be satisfied by I, so it must be the resolvent of two clauses $\beta_1 = \{d_1, \ldots, d_k, a_{i+1}\}$ and $\beta_2 = \{f_1, \ldots, f_j, \neg a_{i+1}\}$ of T^i which resolved upon a_{i+1}. Since I satisfies β_1 and β_2 but not β then it must be the case that I maps a_{i+1} and $\neg a_{i+1}$ into 1, but this contradicts the hypothesis that I is an S-x-interpretation.

(*Only if part for both properties*) Suppose that there is an S-x-interpretation I satisfying T^{i+1} $(0 < i < m)$ then I maps into 1 at least one literal of any clause of T^{i+1}. Let us assume that there is one clause β_1 of T^i which is not satisfied by I we show that this leads to a contradiction. This clause β_1 cannot belong to T^{i+1} because it would be satisfied by I, so in it must occur a_{i+1} and I maps a_{i+1} into 0, or in it occurs $\neg a_{i+1}$ and I maps $\neg a_{i+1}$ into 0. In the first case define a new S-x-interpretation I' which is equivalent to I except that it maps a_{i+1} into 1, notice that this new interpretation I' satisfies T^{i+1} because a_{i+1} does not occur in it. If there is another clause $\beta_2 \in T^i$ which is not satisfied by I' then in β_2 must occur $\neg a_{i+1}$ and now we have that the resolvent of β_1 and β_2 belongs to T^{i+1}, but this resolvent is not satisfied by I' hence contradiction.

We now show that T^m is S-3-satisfiable iff $\square \notin T^m$ and T^m is S-1-satisfiable iff $T^m = \Omega$. Notice that T^m does not contain any of the letters of S, hence T^m is S-3-satisfiable iff it is 3-satisfiable, but any formula not containing $\square$ is 3-satisfiable because the 3-interpretation mapping all the literals into 1 will satisfy it. For the same reason, T^m is S-1-satisfiable iff it is 1-satisfiable, but any formula containing at least one clause is 1-unsatisfiable because the 1-interpretation maps all the literals into 0 and, therefore, does not satisfy it. $\Diamond$

Theorem 5.7. *A formula T is S-3-satisfiable iff there exists a $H \in \Phi^m$ that is S-3-satisfiable; H is S-3-satisfiable iff $\square \notin H$. A formula T is S-1-satisfiable iff there exists an $H \in \Phi^m$ that is S-1-satisfiable; H is S-1-satisfiable iff $H = \Omega$.*

Proof. We divide the proof into two parts, first of all we prove, at the same time, the two equivalences: T is S-3-satisfiable iff there exists a $H \in \Phi^m$ that is S-3-satisfiable and T is S-1-satisfiable iff there exists a $H \in \Phi^m$ that is S-1-satisfiable. In the last part we prove that any $H \in \Phi^m$ is S-3-satisfiable iff $\square \notin H$ and it is H is S-1-satisfiable iff $H = \Omega$. We prove the first part by induction on m. The base case is trivial since $T^0 = T$, in the inductive case we have:

(*If part for both properties*) Suppose that there is an S-x-interpretation I satisfying an $H \in \Phi^i$ $(0 < i < m)$ then I maps into 1 at least one literal of any clause of H. There are two formulae $H_1 = \mathit{simplify}(H, \{a_{i+1}\})$ and $H_2 = \mathit{simplify}(H, \{\neg a_{i+1}\})$ which occur in Φ^{i+1}. Since I is an S-x-interpretation it maps a_{i+1} into 1 and $\neg a_{i+1}$ into 0 or the reverse; we show that in the first case I satisfies H_1, while in the second case I

satisfies H_2. We examine the first case, the other case is similar. We assume that there is a clause β in H_1 which is not satisfied by I and show that this leads to a contradiction. It is clear that β does not belong to H, hence β has been obtained by $\beta_1 = \beta \cup \{\neg a_{i+1}\}$ where $\beta_1 \in H$, but now we have that I does not validate β_1 because I maps into 0 all literals of β and also maps $\neg a_{i+1}$ into 0, hence we obtain a contradiction.

(*Only if part for both properties*) Suppose that there is an S-x-interpretation I satisfying an $H_1 \in \Phi^{i+1}$ $(0 < i < m)$ then I maps into 1 at least one literal of any clause of H_1. Since in H_1 the literals a_{i+1} and $\neg a_{i+1}$ do not occur then also the interpretation I' which is equal to I, except that it maps these two literals into the opposite of the value of I, satisfies H_1. We know that there exist an $H \in \Phi^i$ s.t. either $H_1 = \mathit{simplify}(H, \{a_{i+1}\})$ or $H_1 = \mathit{simplify}(H, \{\neg a_{i+1}\})$, we show that in the first case I satisfies H while in the second case I' satisfies H. We examine the first case, the other case is similar. We assume that there is a clause β in H which is not satisfied by I and show that this leads to a contradiction. It is clear that β has been eliminated in the simplifying process, because otherwise it would be satisfied by I. But the only reason why it may have been eliminated is because in it occurs a_{i+1} and in this case β is satisfied by I, hence contradiction.

The last part is immediately proven noticing that any $H \in \Phi^m$ will not contain any literal of the set S, hence the result trivially follows from the proof of the last part of the Theorem 5.6. ◊

Theorem 5.8. *Let S be the set $\{a_1, \ldots, a_m\}$. T is S-3-unsatisfiable iff $T \models^3 (a_1 \wedge \neg a_1) \vee \cdots \vee (a_m \wedge \neg a_m)$ holds, or equivalently $T \models^3 (c_1 \vee \cdots \vee c_m)$ holds for any combination $\{c_1, \ldots, c_m\}$, where each c_i $(1 \leqslant i \leqslant m)$ is either a_i or $\neg a_i$.*

Proof. See proof of Theorem A.3. ◊

Theorem 5.9. *Let S be the set $\{a_1, \ldots, a_m\}$. T is S-1-satisfiable iff there exists a set $\alpha = \{c_1, \ldots, c_m\}$, where each c_i $(1 \leqslant i \leqslant m)$ is either a_i or $\neg a_i$, such that $\mathit{simplify}(T, \alpha)$ contains no clauses.*

Proof.

(*If part*) Suppose that for a set $\alpha = \{c_1, \ldots, c_m\}$, where each c_i $(1 \leqslant i \leqslant m)$ is either a_i or $\neg a_i$, such that $\mathit{simplify}(T, \alpha)$ does not contain any clause. We define an S-1-interpretation M according to the following rule: M maps every letter l of $L \setminus S$ and its negation $\neg l$ into 0; M maps every letter a_i of $L \setminus S$ into 1 iff a_i occurs in α, otherwise it maps a_i into 0; M maps $\neg a_i$ into the opposite value. Taking into account that the process of simplification deletes all the clauses of T which contain a literal appearing in α, we can easily notice that M maps at least one literal per clause of T into 1. Therefore M is an S-1-interpretation satisfying T.

(*Only if part*) Suppose that T is S-1-satisfiable and let M be an S-1-interpretation satisfying it. We define a set of literals α according to the following rule: if M maps a letter a_i of S into 1 then a_i occurs in α; if M maps a letter a_i of S into 0 then $\neg a_i$ occurs in α. Notice that α is the set $\{c_1, \ldots, c_m\}$, where each c_i $(1 \leqslant i \leqslant m)$ is either a_i or $\neg a_i$. Taking into account that M maps at least one literal per clause of T into 1,

we can easily notice that *simplify*(T, α) does not contain any clause, since the process of simplification deletes all the clauses of T which contain a literal occurring in α. $\Diamond$

A.3. Appendix to Section 6

Theorem 6.1. $(\models \Box_S^3\alpha \rightarrow \Box_S^3\gamma)$ *iff* $(\Box_S^3\alpha \wedge \neg \Box_S^3 \gamma$ *is unsatisfiable*$)$ *iff* $(\alpha \models_S^3 \gamma)$.

Proof. We prove the theorem by showing the following three properties:

(1) ($\Box_S^3\alpha \wedge \neg \Box_S^3 \gamma$ is unsatisfiable) implies $(\models \Box_S^3\alpha \rightarrow \Box_S^3\gamma)$;

(2) $(\models \Box_S^3\alpha \rightarrow \Box_S^3\gamma)$ implies $(\alpha \models_S^3 \gamma)$;

(3) $(\alpha \models_S^3 \gamma)$ implies ($\Box_S^3\alpha \wedge \neg \Box_S^3 \gamma$ is unsatisfiable).

Proof of (1). Assume that $\Box_S^3\alpha \wedge \neg \Box_S^3 \gamma$ is unsatisfiable and $\not\models \Box_S^3\alpha \rightarrow \Box_S^3\gamma$ holds. As a consequence we have:

(1) $\exists \mathcal{M}.\exists s.\mathcal{M}, s \not\models \Box_S^3\alpha \rightarrow \Box_S^3\gamma$.

(2) $\exists \mathcal{M}.\exists s.(\mathcal{M}, s \not\models \neg \Box_S^3 \alpha)$ and $(\mathcal{M}, s \not\models \Box_S^3\gamma)$.

(3) $\exists \mathcal{M}.\exists s.\neg(\exists t \in S\text{-}3(Sit)\ sRt$ and $\mathcal{M}, t \not\models \alpha)$ and $\neg(\forall t \in S\text{-}3(Sit)\ sRt$ implies $\mathcal{M}, t \models \gamma)$.

(4) $\exists \mathcal{M}.\exists s.(\forall t \in S\text{-}3(Sit)\ sRt$ implies $\mathcal{M}, t \models \alpha)$ and $(\exists t \in S\text{-}3(Sit)\ sRt$ and $\mathcal{M}, t \not\models \gamma)$.

(5) $\exists \mathcal{M}.\exists s.\mathcal{M}, s \models \Box_S^3\alpha$ and $\mathcal{M}, s \models \neg \Box_S^3 \gamma$.

(6) $\exists \mathcal{M}.\exists s.\mathcal{M}, s \models \Box_S^3\alpha \wedge \neg \Box_S^3 \gamma$.

(7) $\Box_S^3\alpha \wedge \neg \Box_S^3 \gamma$ is satisfiable, hence contradiction.

Proof of (2). Assume that $(\models \Box_S^3\alpha \rightarrow \Box_S^3\gamma)$ and $(\alpha \not\models_S^3 \gamma)$. Hence, there exists an S-3-interpretation I s.t. $I \models \alpha$ and $I \not\models \gamma$. Let $\mathcal{M} = (Sit, R, V)$ where $Sit = \{s\}$, $R = \{(s, s)\}$ and $V(s) = I$. We have that $\mathcal{M}, s \models \alpha$ and $\mathcal{M}, s \not\models \gamma$. Since s is the only situation in Sit, we also have that $\mathcal{M}, s \models \Box_S^3\alpha$ and $\mathcal{M}, s \models \neg \Box_S^3 \gamma$. Therefore, $\mathcal{M}, s \not\models \Box_S^3\alpha \rightarrow \Box_S^3\gamma$, but this contradicts the assumptions.

Proof of (3). Assume that $\alpha \models_S^3 \gamma$ holds and $\Box_S^3\alpha \wedge \neg \Box_S^3 \gamma$ is satisfiable. As a consequence we have:

(1) $\exists \mathcal{M}.\exists s.\mathcal{M}, s \models \Box_S^3\alpha \wedge \neg \Box_S^3 \gamma$.

(2) $\exists \mathcal{M}.\exists s.(\mathcal{M}, s \models \Box_S^3\alpha)$ and $(\mathcal{M}, s \models \neg \Box_S^3 \gamma)$.

(3) $\exists \mathcal{M}.\exists s.(\forall t \in S\text{-}3(Sit)\ sRt$ implies $\mathcal{M}, t \models \alpha)$ and $(\exists t \in S\text{-}3(Sit)\ sRt$ and $\mathcal{M}, t \not\models \gamma)$.

(4) $\exists \mathcal{M}.\exists s.\exists t \in S\text{-}3(Sit)\ sRt$ and $\mathcal{M}, t \not\models \gamma \wedge \mathcal{M}, t \models \alpha)$.

(5) There exists an x s.t. x is an S-3-interpretation and $x \not\models \alpha$ and $x \models \gamma$.

(6) $\alpha \not\models_S^3 \gamma$, hence contradiction. $\Diamond$

Theorem 6.2. $(\models \Box_S^1\alpha \rightarrow \Box_S^1\gamma)$ *iff* $(\Box_S^1\alpha \wedge \neg \Box_S^1 \gamma$ *is unsatisfiable*$)$ *iff* $(\alpha \models_S^1 \gamma)$.

Proof. Simply replace S-1 for S-3 in the previous proof. $\Diamond$

Theorem 6.3. *Let* $S \subseteq S' \subseteq L$. $\models \Box_S^3\alpha \rightarrow \Box_{S'}^3\alpha$.

Proof. Assume that $\not\models \Box_S^3\alpha \rightarrow \Box_{S'}^3\alpha$. As a consequence we have:

(1) $\exists \mathcal{M}.\exists s.\mathcal{M}, s \not\models \Box^3_S \alpha \rightarrow \Box^3_{S'} \alpha$.
(2) $\exists \mathcal{M}.\exists s.\mathcal{M}, s \models \Box^3_S \alpha$ and $\mathcal{M}, s \not\models \Box^3_{S'} \alpha$.
(3) $\exists \mathcal{M}.\exists s.(\forall t \in S\text{-}3(Sit)\ sRt$ implies $\mathcal{M}, t \models \alpha)$ and $(\exists x \in S'\text{-}3(Sit)\ sRx$ and $\mathcal{M}, x \not\models \alpha)$.
(4) Since $S'\text{-}3(Sit) \subseteq S\text{-}3(Sit)$, we have that $\exists x \in S\text{-}3(Sit)\ sRx$ and $\mathcal{M}, x \not\models \alpha$. But for such an x we also have that $\mathcal{M}, x \models \alpha$, hence contradiction. $\Diamond$

A.4. Appendix to Section 7

Some of the proofs given in the following use the standard semantics of concept languages based on extension functions. Now we briefly recall this semantics (for more details see [68]).

An interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ consists of a set $\Delta^{\mathcal{I}}$ (the domain) and a function $\cdot^{\mathcal{I}}$ that maps every concept to a subset of $\Delta^{\mathcal{I}}$ and every role to a subset of $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ such that

$$\top^{\mathcal{I}} = \Delta^{\mathcal{I}}, \qquad \bot^{\mathcal{I}} = \emptyset,$$

$$(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}, \qquad (\neg A)^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus A^{\mathcal{I}},$$

$$(\forall R.C)^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} \mid \forall b.(a,b) \in R^{\mathcal{I}} \rightarrow b \in C^{\mathcal{I}}\},$$

$$(\exists R.C)^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} \mid \exists b.(a,b) \in R^{\mathcal{I}} \wedge b \in C^{\mathcal{I}}\}.$$

A concept C is satisfiable if and only if there exists an interpretation $\mathcal{I}$ such that $C^{\mathcal{I}}$ is non empty. We say C is subsumed by D ($C \sqsubseteq D$) if for every interpretation $\mathcal{I}$ we have $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$.

In order to prove some of the results of the paper we need the following lemma. Let C be an $\mathcal{ALC}$ concept and D one of its subconcepts, we denote with $C(D/G)$ the concept obtained by replacing every occurrence of D in C with the concept G.

Lemma A.4. *Let C be an $\mathcal{ALC}$ concept and D one of its subconcepts which is not in the scope of any $\neg$ operators. Let G be another $\mathcal{ALC}$ concept. If $D \sqsubseteq G$, then $C(D/G)$ is satisfiable if C is satisfiable. On the other hand, if $G \sqsubseteq D$ then C is satisfiable if $C(D/G)$ is satisfiable.*

Proof. The proof is done by induction on the structure of C. In the base case we have that $C = D$, thus $C(D/G) = G$ and $D \sqsubseteq G$. Then satisfiability of C implies satisfiability of $C(D/G)$. In the general case we have to show that all the language constructors except negation preserve this property. In particular $D \sqsubseteq G$ implies

$$C' \sqcap D \sqsubseteq C' \sqcap G, \qquad C' \sqcup D \sqsubseteq C' \sqcup G,$$

$$\exists R.D \sqsubseteq \exists R.G, \qquad \forall R.D \sqsubseteq \forall R.G.$$

The proof is straightforward for all cases. A dual argument holds for the case of $G \sqsubseteq D$. $\Diamond$

Theorem 7.3. *For each i $(0 \leqslant i \leqslant n+1)$, if $C_i^{\top}$ is unsatisfiable then $C_j^{\top}$ is unsatisfiable for all $j \geqslant i$, hence C is unsatisfiable.*

For each i $(0 \leqslant i \leqslant n+1)$, if $C_i^{\perp}$ is satisfiable then $C_j^{\perp}$ is satisfiable for all $j \geqslant i$, hence C is satisfiable.

Proof. It follows from Lemma A.4. In one case we are replacing subconcepts of C with $\top$ and it is always the case that a concept is subsumed by $\top$. Furthermore, even if $\mathcal{ALE}$ allows negation, we never replace subconcepts which are under the scope of $\neg$ operator so the proof of Lemma A.4 still holds. In the other case, in which we replace subconcepts of C with $\perp$, the other part of Lemma A.4 is used. $\Diamond$

Theorem 7.5. *For all i $(0 \leqslant i \leqslant n+1)$, C is S_i-3-satisfiable iff $C_i^{\top}$ is satisfiable.*

Proof. Dealing with an S-3-interpretation M, when an atom $\alpha \in B_h \setminus S$ belongs both to M^+ and to M^-, we say that α is mapped into *contradiction*.

(Only if part) Suppose that $C_i^{\top}$ is satisfiable, i.e. that $SNF(\Gamma(C_i^{\top}))$ is satisfiable. Let $M = \langle M^+, M^- \rangle$ be an Herbrand model of $SNF(\Gamma(C_i^{\top}))$. We define an S_i-3-interpretation $N = \langle N^+, N^- \rangle$ of $SNF(\Gamma(C))$ according to the following rules:

- for each atom $\alpha \in S_i$:

$$\alpha \in M^+ \Longrightarrow \alpha \in N^+;$$
$$\alpha \in M^- \Longrightarrow \alpha \in N^-;$$

- for each atom $\alpha \in B_h \setminus S_i$: $\alpha \in N^+$ and $\alpha \in N^-$.

Notice that N is necessarily an S_i-3-interpretation of $SNF(\Gamma(C))$, since the Herbrand Base of $SNF(\Gamma(C_i^{\top}))$ is equal to S_i. We now show that N is also an S_i-3-model, thus proving that C is S_i-3-satisfiable. Since $SNF(\Gamma(C))$ is an universally quantified formula, it is sufficient to show that N satisfies all its ground instances. We split the proof in two subcases:

(1) All the variables of $SNF(\Gamma(C))$ are bound to terms belonging to U_i. In this case for every instance g of $SNF(\Gamma(C))$ there exists a corresponding instance h of $SNF(\Gamma(C_i^{\top}))$, where the same variables are bound to the same terms. More precisely, all the atoms of h belong to S_i, and h is obtained from g by substituting some of its subformulae with $\top$. All the atoms occurring in these subformulae are instantiated on terms of $U_k \setminus U_i$, where $k > i$ and U_k is a stratum of the Herbrand Base of $SNF(\Gamma(C))$. Therefore all these atoms belong to $S_k \setminus S_i$, hence are mapped into *contradiction* by N. Since $M \models SNF(\Gamma(C_i^{\top}))$, we know that $M \models h$, hence $N \models h$. Since g differs from h in some literals which are anyway mapped into *contradiction* by N, it follows that $N \models g$.

(2) At least one variable of $SNF(\Gamma(C))$ is bound to a term not belonging to U_i. By definition of N, we know that all the atoms instantiated to those terms are mapped by N to *contradiction*. Since the Herbrand Universe of $SNF(\Gamma(C_i^{\top}))$ is equal to U_i, some instances g of $SNF(\Gamma(C))$ do not have a corresponding instance of $SNF(\Gamma(C_i^{\top}))$. Let us consider an instance g' of $SNF(\Gamma(C_i^{\top}))$ obtained from g by binding any variable not bound to terms belonging to U_i to

terms of U_i in an arbitrary way. From the previous item we know that $N \models g'$. Remember all the atoms instantiated to terms not belonging to U_i are mapped into *contradiction* by N. Therefore g differs from g' in some literals which are mapped into *contradiction* by N, hence it follows that $N \models g$.

(*If part*) Suppose that C is S_i-3-satisfiable, i.e. that $SNF(\Gamma(C))$ is S_i-3-satisfiable. Let $N = \langle N^+, N^- \rangle$ be an S_i-3 Herbrand model of $SNF(\Gamma(C))$. We define an Herbrand interpretation $M = \langle M^+, M^- \rangle$ of $SNF(\Gamma(C_i^\top))$ according to the following rule. For each atom $\alpha \in S_i$:

$$\alpha \in N^+ \Longrightarrow \alpha \in M^+;$$
$$\alpha \in N^- \Longrightarrow \alpha \in M^-;$$

Notice that M is necessarily an Herbrand interpretation of $SNF(\Gamma(C_i^\top))$, since the Herbrand Base of $SNF(\Gamma(C_i^\top))$ is equal to S_i. We now show that M is also an Herbrand model, thus proving that $C_i^\top$ is satisfiable. Since $SNF(\Gamma(C_i^\top))$ is an universally quantified formula, it is sufficient to show that M satisfies all its ground instances g. Let h be any instance of $SNF(\Gamma(C))$ which corresponds to g, where the same variables are bound to the same terms. Since $N \models SNF(\Gamma(C))$, we know that $N \models h$. Moreover h is obtained from g by substituting each occurrence of $\top$ with a formula. Since $\top$ occurs always positively in g and is satisfied by M, it follows that $M \models g$. $\Diamond$

Theorem 7.6. *For all i $(0 \leqslant i \leqslant n+1)$, C is S_i-1-satisfiable iff $C_i^\perp$ is satisfiable.*

Proof. Dealing with an S-1-interpretation M, when an atom $\alpha \in B_h \setminus S$ belongs neither to M^+ nor to M^-, we say that α is mapped into *undefined*.

(*Only if part*) Suppose that $C_i^\perp$ is satisfiable, i.e. that $SNF(\Gamma(C_i^\perp))$ is satisfiable. Let $M = \langle M^+, M^- \rangle$ be an Herbrand model of $SNF(\Gamma(C_i^\perp))$. We define an S_i-1-interpretation $N = \langle N^+, N^- \rangle$ of $SNF(\Gamma(C))$ according to the following rules:

- for each atom $\alpha \in S_i$:

$$\alpha \in M^+ \Longrightarrow \alpha \in N^+;$$
$$\alpha \in M^- \Longrightarrow \alpha \in N^-;$$

- for each atom $\alpha \in B_h \setminus S_i$: $\alpha \notin N^+$ and $\alpha \notin N^-$.

Notice that N is necessarily an S_i-1-interpretation of $SNF(\Gamma(C))$, since the Herbrand Base of $SNF(\Gamma(C_i^\perp))$ is equal to S_i. We now show that N is also an S_i-1-model, thus proving that C is S_i-1-satisfiable. Since $SNF(\Gamma(C))$ is an universally quantified formula, it is sufficient to show that N satisfies all its ground instances. We recall that, by definition of S-1-satisfiability, we have only to consider ground instances of $SNF(\Gamma(C))$ in which variables are substituted by terms of the set $Terms(S_i)$, which are the terms occurring in the set S_i, i.e. are the terms of U_i. Therefore we know that all the variables of $SNF(\Gamma(C))$ are bound to terms belonging to U_i. This implies that for every instance g of $SNF(\Gamma(C))$ there exists a corresponding instance h of $SNF(\Gamma(C_i^\perp))$, where the same variables are bound to the same terms. More precisely, all the atoms of h belong to S_i, and h is obtained from g by substituting some of its subformulae with $\perp$. Since $M \models SNF(\Gamma(C_i^\top))$, we know that $M \models h$, hence $N \models h$. Notice that $\perp$ is not satisfied

by N, therefore $N \models g$ even if the subformulae in which g differs from h are not satisfied. Since this is the "worst" case, it follows that $N \models g$.

(*If part*) Suppose that C is S_i-1-satisfiable, i.e. that $SNF(\Gamma(C))$ is S_i-1-satisfiable. Let $N = \langle N^+, N^- \rangle$ be an S_i-1 Herbrand model of $SNF(\Gamma(C))$. We define an Herbrand interpretation $M = \langle M^+, M^- \rangle$ of $SNF(\Gamma(C_i^\perp))$ according to the following rule. For each atom $\alpha \in S_i$:

$$\alpha \in N^+ \Longrightarrow \alpha \in M^+;$$
$$\alpha \in N^- \Longrightarrow \alpha \in M^-.$$

Notice that M is necessarily an Herbrand interpretation of $SNF(\Gamma(C_i^\perp))$, since the Herbrand Base of $SNF(\Gamma(C_i^\perp))$ is equal to S_i. We now show that M is also an Herbrand model, thus showing that $C_i^\perp$ is satisfiable. Since $SNF(\Gamma(C_i^\perp))$ is an universally quantified formula, it is sufficient to show that M satisfies all its ground instances g. Let h be any instance of $SNF(\Gamma(C))$ which corresponds to g, where the same variables are bound to the same terms. Since $N \models SNF(\Gamma(C))$, we know that $N \models h$. Moreover h is obtained from g by substituting each occurrence of $\perp$ with a formula. All the atoms occurring in this formula are instantiated on terms of $U_k \setminus U_i$, where $k > i$ and U_k is a stratum of the Herbrand Base of $SNF(\Gamma(C))$. Therefore all these atoms belong to $S_k \setminus S_i$, hence are mapped into *undefined* by N. Since M maps $\perp$ into 0, it follows that $M \models g$. $\Diamond$

Theorem 7.7. *Let P be a subset of $\mathcal{A}$ and i be an index such that $0 \leqslant i \leqslant n+1$. If $C_{P,i}^\top$ is unsatisfiable, then any subconcept D of C such that $C_{P,i}^\top \preceq D$ is unsatisfiable. If $C_{P,i}^\perp$ is satisfiable, then any subconcept D of C such that $C_{P,i}^\perp \preceq D$ is satisfiable.*

Proof. The same proof of Theorem 7.3 applies also in this case. In fact, we only allow negation in front of primitive concepts. Hence, we never replace subconcepts that are in the scope of the $\neg$ operator. $\Diamond$

Theorem 7.9. *For all i $(0 \leqslant i \leqslant n+1)$ and $P \subseteq \mathcal{A}$ the concept C is $S_{P,i}$-3-satisfiable iff $C_{P,i}^\top$ is satisfiable.*

Proof. The proof is very similar to that of Theorem 7.5 with the additional complication of replacing some of the primitive concepts and their negation with $\top$. This is however already taken into account when we define the Herbrand Base B_h of a simplified concept $C_{P,i}^\top$. In fact, B_h will only contain atoms of concepts appearing in $C_{P,i}^\top$ and, even in this case, is equal to $S_{P,i}$. Hence, the same proof of Theorem 7.5 applies. $\Diamond$

Theorem 7.10. *For all i $(0 \leqslant i \leqslant n+1)$ and $P \subseteq \mathcal{A}$ the concept C is $S_{P,i}$-1-satisfiable iff $C_{P,i}^\perp$ is satisfiable.*

Proof. The proof is very similar to that of Theorem 7.6 since similar considerations to the ones done in the proof of Theorem 7.9 hold. $\Diamond$

A.5. Appendix to Section 8

Theorem 8.6. *For any S and S' such that $S \subseteq S' \subseteq L$, if α is S-1-Kripke satisfiable, then α is S'-1-Kripke satisfiable (hence 2-Kripke satisfiable). Moreover if α is S-3-Kripke unsatisfiable, then α is S'-3-Kripke unsatisfiable (hence 2-Kripke unsatisfiable).*

Proof. Let α be S'-3-Kripke satisfiable and $\mathcal{M} = \langle W, R, V\rangle$ one of its S'-3-Kripke models. $\mathcal{M}$ is also an S-3-Kripke model, since, for each $w \in W$, $V(w)$ is an S-3-interpretation.

Let α be S-1-Kripke satisfiable and $\mathcal{M} = \langle W, R, V\rangle$ one of its S-1-Kripke models. We build an S'-1-Kripke interpretation $\mathcal{N} = \langle W, R, V'\rangle$ in the following way: for each $w \in W$

- for each $l \in S$, $V'(w)$ maps l into 1 iff $V(w)$ maps l into 1; moreover it maps $\neg l$ into the opposite value;
- for each $l \in S' \setminus S$, $V'(w)$ maps l into 1 and $\neg l$ into 0;
- for each remaining letter l, $V'(w)$ maps l and $\neg l$ into 0.

Notice that for each w, $V'(w)$ is an S'-1-interpretation of L, since it maps every letter l of S' and its negation $\neg l$ into opposite values and it maps each remaining literal into 0. Therefore $\mathcal{N}$ is an S'-1-Kripke interpretation.

Theorem 4.4 (applied when γ is the empty clause) shows that for any propositional formula T and any S, S' such that $S \subseteq S'$, if T is S-1-satisfiable, then T is S'-1-satisfiable. Therefore in each $w \in W$ the set of propositional formulae satisfied by V' is a superset of the set of propositional formulae satisfied by V, hence $\mathcal{N}$ is an S'-1-Kripke model of α. $\Diamond$

Theorem 8.8. *If we restrict our attention to accessibility relations which are reflexive, transitive and euclidean, then there exists one algorithm to decide if α is S-1-Kripke satisfiable and one to decide if α is S-3-Kripke satisfiable both running in $O(m \cdot |\alpha| \cdot 2^{|S|})$ time, where m is the number of occurrences of the modal operator in α.*

Proof. The algorithms for checking S-1- and S-3-Kripke satisfiability are based on a mapping π from any modal formula α on the alphabet L into a propositional formula $\pi(\alpha)$ on the alphabet $\pi(L)$ such that $|\pi(L)| = (m+1) \cdot |L|$ and $|\pi(\alpha)| \leqslant (m+1) \cdot |\alpha|$.

The alphabet $\pi(L)$ is defined as $\bigcup_{i=1}^{m+1} \bigcup_{p \in L} p^i$, that is it contains $m+1$ copies of each letter of L. If S is a subset of L, then $\pi(S)$ is defined as $\bigcup_{i=1}^{m+1} \bigcup_{p \in S} p^i$. The mapping $\pi(\alpha)$ is defined by the following rewriting rules, where $\alpha, \alpha_1, \alpha_2$ are modal formulae, and p is in L:

$$
\begin{aligned}
&\alpha \mapsto (\alpha, 1),\\
&(\alpha_1 \wedge \alpha_2, i) \mapsto (\alpha_1, i) \wedge (\alpha_2, i),\\
&(\alpha_1 \vee \alpha_2, i) \mapsto (\alpha_1, i) \vee (\alpha_2, i),\\
&(\neg(\alpha_1 \wedge \alpha_2), i) \mapsto (\neg\alpha_1, i) \vee (\neg\alpha_2, i),\\
&(\neg(\alpha_1 \vee \alpha_2), i) \mapsto (\neg\alpha_1, i) \wedge (\neg\alpha_2, i),\\
&(\boldsymbol{K}\alpha, i) \mapsto (\alpha, 1) \wedge \cdots \wedge (\alpha, m+1),\\
&(\neg\boldsymbol{K}\alpha, i) \mapsto (\neg\alpha, 1) \vee \cdots \vee (\neg\alpha, m+1),
\end{aligned}
$$

$(p, i) \mapsto p^i$,
$(\neg p, i) \mapsto \neg p^i$.

It is easy to prove that the S-1-Kripke satisfiability of the modal formula α is equivalent to the S'-1-satisfiability of the propositional formula $\pi(\alpha)$, where $S' = \pi(S)$. In fact the mapping π is based on a generalization of a property of the system $S5$, stating that if β is a 2-Kripke satisfiable formula with m occurrences of the modal operator, then there exists a 2-Kripke interpretation $\mathcal{M} = \langle W, R, V \rangle$ and a $w \in W$ such that $\mathcal{M}, w \models \beta$ and where the size of W is less than $m + 1$ (see [52, Lemma 6.1]).

The S'-1-satisfiability of $\pi(\alpha)$ can be determined in $O(m \cdot |\alpha| \cdot 2^{|S|})$ time with the algorithms presented in Appendix A.1. Analogous properties hold for the S-3-Kripke satisfiability of α. ◊

Theorem 8.9. *If the accessibility relation is unrestricted, then deciding if α is S-1-Kripke satisfiable and deciding if α is S-3-Kripke satisfiable are PSPACE-complete problems even if $|S| = 1$.*

Proof. It is proven in [67] that any formula of the $\mathcal{K}$ system can be polynomially mapped in a concept of the language $\mathcal{ALC}$ such that satisfiability is preserved. It is proven in [68] that satisfiability of an $\mathcal{ALC}$ concept is a PSPACE-complete problem, even if only one primitive concept is used. As a consequence, satisfiability of a formula of the $\mathcal{K}$ system in which only one propositional letter occur is PSPACE-complete. ◊

Theorem 8.10. *Let α be a modal formula and X be a modal system admitting the K schema and the rule of necessitation. The formula α is $\langle S, i \rangle$-3-Kripke satisfiable in the system X iff $\psi^3_{S,i}(\alpha)$ is 2-Kripke satisfiable in the system X.*

Proof. (*Only if part*) Assume that α is $\langle S, i \rangle$-3-Kripke satisfiable. Hence, there exists a model (over the alphabet L) $\mathcal{M} = \langle W, R, V \rangle$ and a world $w \in W$ s.t. $\mathcal{M}, w \models^3_{S,i} \alpha$. Let $\mathcal{M}_1 = \langle W_1, R_1, V_1 \rangle$ be a new model (over the alphabet S) where $W_1 = W$, $R_1 = R$, $V_1(x) = V(x)$ and $V_1(\neg x) = V(\neg x)$ if $x \in S$. Notice that $\mathcal{M}_1$ is a 2-Kripke interpretation.

We prove that $\mathcal{M}_1, w \models \psi^3_{S,i}(\alpha)$ by a double induction on i and the size of α. When i is smaller than 0 $\psi^3_{S,i}(\alpha) = t$ and, therefore, $\mathcal{M}_1, w \models \psi^3_{S,i}(\alpha)$. If $\alpha = p$ or $\alpha = \neg p$ it is clearly the case that $\mathcal{M}_1, w \models \psi^3_{S,i}(\alpha)$. If $\alpha = \alpha_1 \wedge (\vee)\alpha_2$ then $\mathcal{M}_1, w \models \psi^3_{S,i}(\alpha)$ iff $\mathcal{M}_1, w \models \psi^3_{S,i}(\alpha_1)$ and (or) $\mathcal{M}_1, w \models \psi^3_{S,i}(\alpha_2)$.

If $\alpha = \boldsymbol{K}\beta$, then it is the case that $\forall t.wRt$ implies $\mathcal{M}, t \models^3_{S,i-1} \beta$. By the inductive hypothesis, $\mathcal{M}, t \models^3_{S,i-1} \beta$ implies $\mathcal{M}_1, t \models \psi^3_{S,i-1}(\beta)$, hence $\forall t.wRt$ implies $\mathcal{M}_1, t \models \psi^3_{S,i-1}(\beta)$. Therefore, we also have that $\mathcal{M}_1, w \models \boldsymbol{K}\psi^3_{S,i-1}(\beta)$. Since it is easy to show that $\psi^3_{S,i}(\alpha) = \boldsymbol{K}\psi^3_{S,i-1}(\beta)$, we also have $\mathcal{M}_1, w \models \psi^3_{S,i}(\alpha)$.

A similar proof also applies to $\alpha = \neg\boldsymbol{K}\beta$.

(*If part*) Assume that $\psi^3_{S,i}(\alpha)$ is satisfiable. Hence, there exists a model, over the alphabet S, $\mathcal{M}_1 = \langle W_1, R_1, V_1 \rangle$ and a world $w \in W_1$ s.t. $\mathcal{M}, w \models \psi^3_{S,i}(\alpha)$. Let $\mathcal{M} =$

$\langle W, R, V\rangle$ be a new model over the alphabet L, where $W = W_1$, $R = R_1$, $V(x) = V_1(x)$ and $V(\neg x) = V_1(\neg x)$ if $x \in S$ and $V(x) = V(\neg x) = 1$ if $x \in L \setminus S$. Notice that $\mathcal{M}$ is an S-3-Kripke interpretation.

We prove that $\mathcal{M}, w \models^3_{S,i} \alpha$ by a double induction on i and the size of α. When i is smaller than 0, $\mathcal{M}, w \models^3_{S,i} \alpha$ holds by definition. If $\alpha = p$ or $\alpha = \neg p$ it is clearly the case that $\mathcal{M}, w \models^3_{S,i} \alpha$. If $\alpha = \alpha_1 \wedge (\vee)\alpha_2$ then $\mathcal{M}, w \models^3_{S,i} \alpha$ iff $\mathcal{M}, w \models^3_{S,i} \alpha_1$ and (or) $\mathcal{M}, w \models^3_{S,i} \alpha_2$.

If $\alpha = \boldsymbol{K}\beta$, then it is the case that $\forall t.wRt$ implies $\mathcal{M}_1, t \models \psi^3_{S,i-1}(\beta)$. By the inductive hypothesis, $\mathcal{M}_1, t \models \psi^3_{S,i-1}(\beta)$ implies $\mathcal{M}, t \models^3_{S,i-1} \beta$, hence $\forall t.wRt$ implies $\mathcal{M}, t \models^3_{S,i-1} \beta$. Therefore, we also have that $\mathcal{M}, w \models^3_{S,i-1} \boldsymbol{K}\beta$ which in turn implies that $\mathcal{M}, w \models^3_{S,i} \alpha$.

A similar proof also applies to $\alpha = \neg\boldsymbol{K}\beta$. $\Diamond$

References

[1] H. Aït-Kaci and R. Nasr, Login: a logic programming language with built-in inheritance, *J. Logic Program.* **3** (1986) 185–215.

[2] A. Anderson and N. Belnap, *Entailment: the Logic of Relevance and Necessity* (Princeton University Press, Princeton, NJ, 1975).

[3] C. Beeri, Data models and languages for databases, in: *Proceedings International Conference on Database Theory (ICDT-88)* (1988) 19–40.

[4] A. Borgida R.J. Brachman, D.L. McGuinness and L.A. Resnick, CLASSIC: a structural data model for objects, in: *Proceedings ACM SIGMOD International Conference on the Management of Data* (1989) 58–67.

[5] R.J. Brachman, "Reducing" CLASSIC to practice: knowledge representation theory meets reality, in: *Proceedings Third International Conference on the Principles of Knowledge Representation and Reasoning (KR-92)*, Cambridge, MA (1992)247–258.

[6] R.J. Brachman and H.J. Levesque, A fundamental tradeoff in knowledge representation and reasoning, in: R.J. Brachman and H.J. Levesque, eds., *Readings in Knowledge Representation* (Morgan Kaufmann, San Mateo, CA, 1985) 41–70.

[7] R.J. Brachman and J. Schmolze, An overview of the KL-ONE knowledge representation system, *Cogn. Sci.* **9** (1985) 171–216.

[8] T. Bylander, The monotonic abduction problem: A functional characterization on the edge of tractability, in: *Proceedings Second International Conference on the Principles of Knowledge Representation and Reasoning (KR-91)*, Cambridge, MA (1991) 70–77.

[9] M. Cadoli, Semantical and computational aspects of Horn approximations, in: *Proceedings IJCAI-93*, Chambery, France (1993) 39–44.

[10] M. Cadoli and M. Lenzerini, The complexity of propositional closed world reasoning and circumscription, *J. Comput. Syst. Sci.* **48** (1994) 255–310.

[11] M. Cadoli and M. Schaerf, Approximate entailment, in: *Trends in Artificial Intelligence: Proceedings 2nd Conference of the Italian Association for Artificial Intelligence (AI*IA-91)*, Lecture Notes In Artificial Intelligence **549** (Springer-Verlag, Berlin, 1991) 68–77.

[12] M. Cadoli and M. Schaerf, Approximate reasoning and non-omniscient agents, in: *Proceedings Fourth Conference on Theoretical Aspects of Reasoning about Knowledge (TARK-92)* (1992) 169–183.

[13] M. Cadoli and M. Schaerf, Approximation in concept description languages, in: B. Nebel, C. Rich and W. Swartout, eds., *Proceedings Third International Conference on the Principles of Knowledge Representation and Reasoning (KR-92)* (1992) 330–341.

[14] M. Cadoli and M. Schaerf, Approximate inference in default reasoning and circumscription, *Fund. Informaticae* (to appear); Preliminary version in: *Proceedings Tenth European Conference on Artificial Intelligence (ECAI-92)*, Vienna, Austria (1992) 319–323.

[15] M. Cadoli and M. Schaerf, The complexity of entailment in propositional multivalued logics, *Annals of Mathematics and Artificial Intelligence* (to appear); preliminary version in: *Proceedings AAAI Spring Symposium Series Workshop on AI and NP-hard problems* (1993) 15–21.

[16] B. Chellas, *Modal Logic: An Introduction* (Cambridge University Press, Cambridge, England, 1980).

[17] S.A. Cook, The complexity of theorem-proving procedures, in: *Proceedings Third ACM Symposium on Theory of Computing*, New York (1971) 151–158.

[18] P. Cousot and R. Cousot, Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints, in: *Proceedings Fourth ACM Symposium on Principles of Programming Languages (POPL-77)*, Los Angeles, CA (1977) 238–252.

[19] J.M. Crawford and B.J. Kuipers, Towards a theory of access limited reasoning, in: *Proceedings First International Conference on the Principles of Knowledge Representation and Reasoning (KR-89)*, Toronto, Ont. (1989) 67–78.

[20] M. Dalal, Tractable instances of some hard deduction problems, in: *Proceedings Tenth European Conference on Artificial Intelligence (ECAI-92)*, Vienna, Austria (1992) 354–358.

[21] M. Dalal and D.W. Etherington, A hierarchy of tractable satisfiability problems, *Inf. Process. Lett.* **44** (1992) 173–180.

[22] M. Davis and H. Putnam, A computing procedure for quantification theory, *J. ACM* **7** (1960) 201–215.

[23] T. Dean and M. Boddy, An analysis of time-dependent planning, in: *Proceedings AAAI-88*, St. Paul, MN (1988) 49–54.

[24] R. Dechter and J. Pearl, Network-based heuristics for constraint-satisfaction problems, *Artif. Intell.* **34** (1988) 1–38.

[25] F.M. Donini, B. Hollunder, M. Lenzerini, A. Marchetti Spaccamela, D. Nardi and W. Nutt, The complexity of existential quantification in concept languages, *Artif. Intell.* **53** (1992) 309–327.

[26] F.M. Donini, M. Lenzerini, D. Nardi and W. Nutt, The complexity of concept languages, in: *Proceedings Second International Conference on the Principles of Knowledge Representation and Reasoning (KR-91)*, Cambridge, MA (Morgan Kaufmann, San Mateo, CA, 1991) 151–162.

[27] F.M. Donini, M. Lenzerini, D. Nardi and W. Nutt, Tractable concept languages, in: *Proceedings IJCAI-91*, Sydney, Australia (1991) 458–463.

[28] W.P. Dowling and J.H. Gallier, Linear-time algorithms for testing the satisfiability of propositional Horn formulae, *J. Logic Program.* **1** (1984) 267–284.

[29] B. Dreben and W.D. Goldfarb, *The Decision Problem. Solvable Classes of Quantificational Formulas* (Addison-Wesley, Reading, MA, 1979).

[30] M. Dunn, Intuitive semantics for first-degree entailments and 'coupled trees', *Philos. Stud.* **29** (1976) 149–168.

[31] T. Eiter and G. Gottlob, The complexity of logic-based abduction, in: *Proceedings 10th Symposium on Theoretical Aspects of Computer Science*, Lecture Notes in Computer Science **665** (Springer-Verlag, Berlin, 1993).

[32] D.V. Etherington and J.M. Crawford, Towards efficient default reasoning, Presented at the 4th International Workshop on Nonmonotonic Reasoning (1992); Notes edited by H. Kautz and D.W. Etherington (1992).

[33] R. Fagin and J.Y. Halpern, Belief, awareness and limited reasoning, *Artif. Intell.* **34** (1988) 39–76.

[34] R. Fagin, J.Y. Halpern and M.Y. Vardi, A nonstandard approach to the logical omniscience problem, in: *Proceedings Third Conference on Theoretical Aspects of Reasoning about Knowledge (TARK-90)*, Pacific Grove, CA (1990) 41–55.

[35] A.M. Frisch, Using model theory to specify AI programs, in: *Proceedings IJCAI-85*, Los Angeles, CA (1985) 148–154.

[36] A.M. Frisch, Inference without chaining, in: *Proceedings IJCAI-87*, Milan, Italy (1987) 515–519.

[37] G. Gallo and M.G. Scutellà, Polynomially solvable satisfiability problems, *Inf. Process. Lett.* **29** (1988) 221–227.

[38] M.L. Ginsberg, Anytime declarativism, Tech. Rept., Stanford University, Depatment of Computer Science, Stanford, CA (1991).

[39] M.L. Ginsberg, Computational considerations in reasoning about action, in: *Proceedings Second International Conference on the Principles of Knowledge Representation and Reasoning (KR-91)*, Cambridge, MA (1991) 250–261.
[40] F. Giunchiglia and T. Walsh, A theory of abstraction, *Artif. Intell.* **57** (1992) 323–389.
[41] R. Greiner and D. Schuurmans, Learning useful Horn approximations, in: *Proceedings Third International Conference on the Principles of Knowledge Representation and Reasoning (KR-92)*, Cambridge, MA (1992) 383–392.
[42] J.Y. Halpern, Reasoning about knowledge: a survey circa 1991, in: A.A. Kent and J.G. Williams, eds., *Encyclopedia of Computer Science and Technology* (Marcel Dekker, New York, 1991).
[43] J. Hintikka, *Knowledge and Belief* (Cornell University Press, Ithaca, NY, 1962).
[44] T. Imielinski, Domain abstraction and limited reasoning, in: *Proceedings IJCAI-87*, Milan, Italy (1987) 997–1003.
[45] D. Israel, The role of logic in knowledge representation, *IEEE Computer* **16** (1983) 37–42.
[46] R.G. Jeroslow and J. Wang, Solving propositional satisfiability problems, *Ann. Math. Artif. Intell.* **1** (1990) 167–187.
[47] H.A. Kautz and B. Selman, A general framework for knowledge compilation, in: H. Richter and M. Richter, eds., *Proceedings of International Workshop on Processing Declarative Knowledge (PDK-91)*, Lecture Notes in Artificial Intelligence **567** (Springer Verlag, Berlin, 1991) 287–300.
[48] H.A. Kautz and B. Selman, Hard problems for simple default logics, *Artif. Intell.* **49** (1991) 243–279.
[49] H.A. Kautz and B. Selman, Forming concepts for fast inference, in: *Proceedings AAAI-92*, San Jose, CA (1992) 786–793.
[50] S.C. Kleene, *Mathematical Logic* (Wiley, New York, 1966).
[51] S.A. Kripke, Semantical considerations on modal logic, *Acta Philos. Fenn.* **16** (1963) 83–94.
[52] R. Ladner, The computational complexity of provability in systems of modal propositional logic, *SIAM J. Computing* **6** (1977) 467–480.
[53] G. Lakemeyer, Tractable meta-reasoning in propositional logics of belief, in: *Proceedings IJCAI-87*, Milan, Italy (1987) 402–408.
[54] H.J. Levesque, A logic of implicit and explicit belief, in: *Proceedings AAAI-84*, Austin, TX (1984) 198–202.
[55] H.J. Levesque, Logic and the complexity of reasoning, *J. Philos. Logic* **17** (1988) 355–389.
[56] H.J. Levesque, A knowledge-level account of abduction, in: *Proceedings IJCAI-89*, Detroit, MI (1989) 1061–1067.
[57] D. Loveland, *Automated Theorem Proving: A Logical Basis* (North-Holland, Amsterdam, 1978).
[58] R.M. MacGregor, Inside the LOOM desscription classifier, *SIGART Bull.* **2** (1991) 88–92.
[59] B. Nebel, Terminological reasoning is inherently intractable, *Artif. Intell.* **43** (1990) 235–249.
[60] N.J. Nilsson, Logic and artificial intelligence, *Artif. Intell.* **47** (1991) 31–56.
[61] P.F. Patel-Schneider, A four-valued semantics for terminological logic, *Artif. Intell.* **38** (1989) 319–351.
[62] P.F. Patel-Schneider, A decidable first-order logic for knowledge representation, *J. Automated Reason.* **6** (1990) 361–388.
[63] J.A. Robinson, A machine oriented logic based on the resolution principle, *J. ACM* **12** (1965) 397–415.
[64] H. Rogers, *Theory of Recursive Functions and Effective Computability* (McGraw-Hill, New York, 1967).
[65] S.J. Russell and S. Zilberstein, Composing real-time systems, in: *Proceedings IJCAI-91*, Sydney, Australia (1991) 212–217.
[66] T. Schaefer, The complexity of satisfiability problems, in: *Proceedings 10th ACM Symposium on Theory of Computing* (1978) 216–226.
[67] K. Schild, A correspondence theory for terminological logics: preliminary report, in: *Proceedings IJCAI-91*, Sydney, Australia (1991) 466–471.
[68] M. Schmidt-Schauß and G. Smolka, Attributive concept descriptions with complements, *Artif. Intell.* **48** (1) (1991) 1–26.
[69] B. Selman and H.A. Kautz, Knowledge compilation using Horn approximations, in: *Proceedings AAAI-91*, Anaheim, CA (1991).
[70] B. Selman and H.J. Levesque, The tractability of path-based inheritance, in: *Proceedings IJCAI-89*, Detroit, MI (1989).

[71] J. Stillman, It's not my default: The complexity of membership problems in restricted propositional default logics, in: *Proceedings AAAI-90*, Boston, MA (1990).

[72] A. Van Gelder, A satisfiability tester for non-clausal propositional calculus, *Inf. Comput.* **79**(1) (1988) 1–21.

[73] M.Y. Vardi, Querying logical databases, *J. Comput. Syst. Sci.* **33** (1986) 142–160.