

A Logic for Anytime Deduction and Anytime Compilation

Frédéric Koriche

LIRMM, UMR 5506, Université Montpellier II CNRS
161, rue Ada, 34392 Montpellier Cedex 5, France
email: koriche@lirmm.fr

Abstract. One of the main characteristics of logical reasoning in knowledge based systems is its high computational complexity. Anytime deduction and anytime compilation are two attractive approaches that have been proposed for addressing such a difficulty. The first one offers a compromise between the time complexity needed to compute approximate answers and the quality of these answers. The second one proposes a trade-off between the space complexity of the compiled knowledge base and the number of possible answers that can be efficiently processed by this data structure. The purpose of this paper is to define a logic which handles these two approaches by incorporating several major features. First, the logic is semantically founded on the notion of resource which determines both the accuracy and the cost of approximation. Second, a stepwise procedure is included for improving approximate answers and allowing their convergence to the correct answer. Third, both sound approximations and complete ones are covered. Fourth and finally, the reasoning task may be done off-line and compiled theories can be used for answering many queries. This logic is applied to the specifications of anytime deducers and anytime compilers.

Topic areas: Foundations of knowledge-based systems, automated reasoning. **Keywords:** Reasoning about knowledge, knowledge compilation, approximate reasoning, anytime deducers, anytime compilers.

1 Introduction

During these past decades, the problem of reasoning about commonsense knowledge has received a great deal of attention in the artificial intelligence community. A widely accepted framework for studying this issue is the knowledge based system approach [14]. Knowledge is described in some logical formalism, called the *representation language*, and stored in a knowledge base. This component is combined with a reasoning mechanism, which is used to determine whether a given sentence, assumed to capture the query, is entailed from the knowledge base. However, it is well known that deduction is very much demanding from a computational point of view. In particular, if the knowledge base and the query are represented in propositional logic, then checking whether the query is entailed from the knowledge base or not is a coNP-complete problem, that is, a problem which probably requires exponential time to be solved.

Anytime deduction and *anytime compilation* are two attractive approaches that have been proposed in propositional logic for addressing such difficulties.

In the first approach, the goal is to define a family of entailment relations that “approximate” classical entailment, by relaxing soundness or completeness of reasoning. The knowledge based system can provide partial solutions even if stopped prematurely; the accuracy of the solution improves with the time used in computing the solution. Hence, anytime deduction offers a compromise between the time complexity needed to compute answers by means of approximate entailment relations and the quality of these answers. Following this idea, Dalal in [5] presents a general technique for approximating deduction problems. The starting point of its framework relies on the entailment relation \vdash^{BCP} defined using boolean constraint propagation. Based on this relation, the author defines a family of entailment relations \vdash_k^{BCP} , which extend \vdash^{BCP} by allowing chaining on sentences of size k . Each relation \vdash_k^{BCP} is sound but incomplete with respect to classical entailment. A different method has been proposed by Cadoli and Schaerf in [2,17]. Their framework includes a parameter S , a set of atomic propositions, which captures the quality of approximation. Based on this parameter, the authors define two families of entailment relations, named \vdash_1^S and \vdash_3^S , which are respectively unsound but complete and sound but incomplete with respect to classical entailment. Several extensions of this framework have been proposed in the domains of non-monotonic logics [3] diagnostic reasoning [22], and reasoning in presence of inconsistency [9,10].

The second approach is concerned by preprocessing a knowledge base into an appropriate data structure which is used for query answering. The goal here is to invest computational resources in the preprocessing effort which will later substantially speed up query answering, in the expectation that the cost of compilation will be amortized over many queries. Compilation is called “exact” if the data structure is logically equivalent to the initial knowledge base, thus guaranteeing answers to all possible queries (see e.g. [16,23]). However, in exact compilation it has been observed that the compiled knowledge base often occupies space exponential in the size of the initial source [19]. This undesirable effect has lead several researchers to explore the possibility of compiling the knowledge base into a family of data structures that “approximate” the initial knowledge base, giving up soundness or completeness of reasoning. The system attempts to compile a knowledge base exactly until a given resource limit is reached, and may answer queries before the completion of compilation. One can view anytime compilation as a technique which offers a trade-off between the space complexity of the compiled knowledge base and the number of queries that can be efficiently processed by this data structure. For example, several authors present anytime methods based on prime implicates generation which are sound but incomplete with respect to exact compilation [7,13,15]. Dually, Schrag in [18] proposes a prime implicants generation algorithm which is unsound but complete with respect to exact compilation. An analogous strategy has been proposed by Selman and Kautz in the context of “Horn approximation” for computing all the greatest lower bounds (GLB) of a clausal knowledge base [20].

The purpose of the paper is to introduce a unifying, logic oriented framework that captures the main ideas of these two approaches. Our investigation generalizes and expands in several directions previous results by Cadoli and Schaerf in [2,17]. The framework is based on multi-modal logic which contains a well-founded semantics and a correct and complete axiomatization. Moreover, the framework integrates the following major features :

- The logic is semantically founded on the notion of *resource* which reflects both the accuracy and the computational cost of the approximations.
- The framework enables *incremental reasoning*: the quality of approximations is a nondecreasing function of the resources that have been spent. Hence, approximate answers can be improved and may converge to the right answer.
- The framework covers *dual reasoning*: both sound but incomplete and complete but unsound answers are returned at any step; they respectively correspond to the lower and upper bounds of the range of possible conclusions that approximate the right answer.
- The framework allows *off-line reasoning* : the knowledge base can be compiled and the resulting data structure may be used for efficiently processing a large set of queries.

The formalism we propose is flexible enough to be applied to several anytime reasoning methods. In this study, we concentrate on the specifications of *anytime deducers* and *anytime compilers* which extend the traditional notion of knowledge based systems. Anytime deducers incorporate the first three properties of our framework ; they approximate the reasoning task by iteratively increasing their inference capabilities. Anytime compilers also exploit off-line reasoning by iteratively computing better and better approximations of their knowledge base.

The rest of the paper is organized as follows. Section 2 formally defines the syntax, the semantics, and a sound and complete axiomatization for the logic. Sections 3 and 4 are devoted to the formal specifications of anytime deducers and anytime compilers. Finally, section 5 suggests some topics for future research. The proof of soundness and completeness of the logic is left in Appendix A.

2 The logic

In this section, we present a propositional logic, named **ARL**, for anytime reasoning. We insist on the fact that the logic is being used here as a specification tool to describe an anytime reasoner rather than as a calculus to be used by one. We begin to define the syntax of the logic, next we examine its semantics in detail, and then we present a sound and complete axiomatization for **ARL**.

2.1 Syntax

In this study, we consider a propositional language constructed from a finite set of atomic propositions (atoms for short) P . In order to formalize the reasoning capabilities of a knowledge based system, we model the notion of deductive

inference as an exploration in a space of possibilities. In a propositional setting, this space is defined by the collection of all the interpretations defined from the atoms of P . Following [17], the notion of *resource* is captured by a parameter S , a subset of P , which corresponds to a limited exploration in this space.

The main contribution of this logic relies on two families of modalities \Box_S and \Diamond_S , defined for each subset S of P . The operator \Box_S is to capture sound but incomplete inference and \Diamond_S to capture complete but unsound inference. Based on these considerations, the language of **ARL** is defined by the smallest set of *sentences* built from the following rules: if p is an atom of P then p is a sentence, if α is a sentence, then $\neg\alpha$ is a sentence, if α and β are sentences then $\alpha \wedge \beta$ and $\alpha \vee \beta$ are sentences, and finally, if α is a sentence that does not contain any occurrence of the modalities \Box_S and \Diamond_S , then $\Box_S \alpha$ and $\Diamond_S \alpha$ are sentences. We remark that the syntax does not allow nested modal operators. A sentence such as $\Box_S \alpha$ is read “the system necessarily infers α given the resources S ”; dually $\Diamond_S \alpha$ is read “the system possibly infers α given the resources S ”.

Other connectives \supset and \equiv are defined in terms of \neg , \wedge and \vee ; that is, $\alpha \supset \beta$ is an abbreviation of $\neg\alpha \vee \beta$ and $\alpha \equiv \beta$ is an abbreviation of $(\alpha \supset \beta) \wedge (\beta \supset \alpha)$. A *declaration* is a sentence without any occurrence of modalities \Box_S and \Diamond_S , and a *knowledge base* A is a finite conjunction of declarations. When there is no risk of confusion, we shall model knowledge bases as sets of declarations.

2.2 Semantics

The basic building block of the semantics is a domain T of truth values which determines the interpretation of sentences and the properties of logical consequence. In the context of limited reasoning, the four valued semantics first proposed by Belnap [1] and Dunn [8], and notably studied in [12] meets our needs. It is a simple modification of classical interpretation in which sentences take as truth-values subsets of $\{0, 1\}$, instead simply either 0 or 1 alone. So, in the logic **ARL**, sentences can be valued to be true, false, both, or neither.

Based on this structure, we define a *valuation* as a total function v from P to T . The space of valuations generated from P is denoted \mathcal{V}_P . A *possible world* is a valuation which maps every atom p of P into $\{1\}$ or $\{0\}$. The space of possible worlds generated from P is denoted \mathcal{W}_P . We say that a valuation v is more *specific* than v' and write $v \subseteq v'$, if for any atom $p \in P$, $v(p) \subseteq v'(p)$ holds.

The concept of approximation is semantically represented by an equivalence relation between valuations. Given a resource parameter S , we say that two valuations v and v' are *S-equivalent* and write $v \sim_S v'$, if and only if for every atom $p \in P$, if $p \in S$ then $v(p) = v'(p)$. It is easy to prove that \sim_S is indeed a reflexive, symmetric and transitive relation. Intuitively, a relation of *S-equivalence* induces a partition of the set \mathcal{V}_P into equivalence classes whose granularity captures the accuracy of approximation. When the resource parameter increases, the partition becomes “finer” and the approximation more precise. The “coarsest” partition is obtained when S is the empty set; in this case, \sim_S is the total relation over \mathcal{V}_P . Conversely, the “finest” partition is given when S is the set P ; in this case \sim_S is the identity relation over \mathcal{V}_P .

The figure 1 illustrates a space of valuations and a relation of S -equivalence defined for $P = \{p, q\}$ and $S = \{p\}$. The nodes and the edges represent the valuations and the induced inclusion relation defined from T . The sets of nodes connected by bold edges represent the equivalence classes.

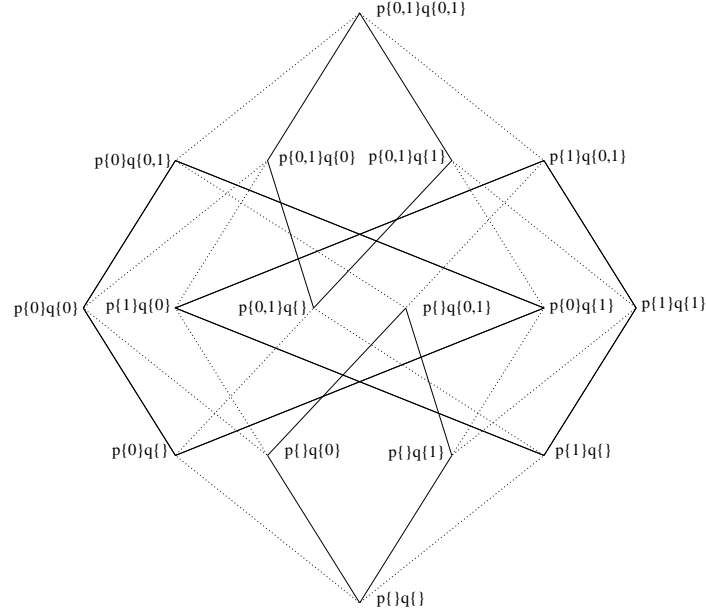


Figure 1. A space of valuations and a relation of S -equivalence.

With these notions in hand, we can now define the semantics of our logic. An *interpretation* of **ARL** consists of a *truth support relation* \models_1 and a *falsity support relation* \models_0 inductively defined by the following conditions:

- (1) $v \models_1 p$ iff $1 \in v(p)$,
 $v \models_0 p$ iff $0 \in v(p)$,
- (2) $v \models_1 \neg\alpha$ iff $v \models_0 \alpha$,
 $v \models_0 \neg\alpha$ iff $v \models_1 \alpha$,
- (3) $v \models_1 \alpha \wedge \beta$ iff $v \models_1 \alpha$ and $v \models_1 \beta$,
 $v \models_0 \alpha \wedge \beta$ iff $v \models_0 \alpha$ or $v \models_0 \beta$,
- (4) $v \models_1 \alpha \vee \beta$ iff $v \models_1 \alpha$ or $v \models_1 \beta$,
 $v \models_0 \alpha \vee \beta$ iff $v \models_0 \alpha$ and $v \models_0 \beta$,
- (5) $v \models_1 \Box_S \alpha$ iff $\forall v' \in \mathcal{V}_P$, if $v \sim_S v'$ then $v' \models_1 \alpha$,
 $v \models_0 \Box_S \alpha$ iff $v \not\models_1 \Box_S \alpha$,
- (6) $v \models_1 \Diamond_S \alpha$ iff $\exists v' \in \mathcal{V}_P$ such that $v \sim_S v'$ and $v' \models_1 \alpha$,
 $v \models_0 \Diamond_S \alpha$ iff $v \not\models_1 \Diamond_S \alpha$.

A sentence α is called *satisfiable* if and only if there exists a possible world $w \in \mathcal{W}_P$ such that $w \models_1 \alpha$. We say that a sentence α is *valid* and write $\models \alpha$, if and only if, for every possible world $w \in \mathcal{W}_P$, $w \models_1 \alpha$ holds. Finally, given two sentences α and β , we say that β is a *logical consequence* of α if and only if $\models \alpha \supset \beta$ holds. The following lemma captures an important structural property of the support relations. It will be frequently used in the remaining sections.

Lemma 1. *For any declaration α and any pair of valuations v, v' such that $v \subseteq v'$, if $v \models_1 \alpha$ then $v' \models_1 \alpha$, and if $v \models_0 \alpha$ then $v' \models_0 \alpha$.*

Proof. Straightforward by induction on the structure of α .

2.3 Axiomatization

We now focus on obtaining a sound and complete axiomatization for our logic. An *axiom system* consists of a collection of *axioms* and *inferences rules*. A *proof* in an axiom system is a finite sequence of sentences, each of which is either an instance of an axiom or follows by an application of an inference rule. Finally, we say that a sentence α is a *theorem* of the axiom system and write $\vdash \alpha$ if there exists a proof of α in the system. The axiom system of **ARL** is the following :

(A1) All tautologies of propositional logic.

(A2) $\Box_S \neg\neg\alpha \equiv \Box_S \alpha$

(A3) $\Box_S (\alpha \wedge \beta) \equiv \Box_S (\beta \wedge \alpha)$
 $\Box_S (\alpha \vee \beta) \equiv \Box_S (\beta \vee \alpha)$

(A4) $\Box_S (\alpha \wedge (\beta \wedge \gamma)) \equiv \Box_S ((\alpha \wedge \beta) \wedge \gamma)$
 $\Box_S (\alpha \vee (\beta \vee \gamma)) \equiv \Box_S ((\alpha \vee \beta) \vee \gamma)$

(A5) $\Box_S (\alpha \wedge (\beta \vee \gamma)) \equiv \Box_S ((\alpha \wedge \beta) \vee (\alpha \wedge \gamma))$
 $\Box_S (\alpha \vee (\beta \wedge \gamma)) \equiv \Box_S ((\alpha \vee \beta) \wedge (\alpha \vee \gamma))$

(A6) $\Box_S \neg(\alpha \wedge \beta) \equiv \Box_S (\neg\alpha \vee \neg\beta)$
 $\Box_S \neg(\alpha \vee \beta) \equiv \Box_S (\neg\alpha \wedge \neg\beta)$

(A7) $\Box_S \alpha \wedge \Box_S \beta \equiv \Box_S (\alpha \wedge \beta)$
 $\Box_S \alpha \vee \Box_S \beta \equiv \Box_S (\alpha \vee \beta)$

(A8) $\Box_{S \cup \{p\}} (p \vee \neg p)$
 $\Diamond_{S - \{p\}} (p \wedge \neg p)$

(A9) $\Box_S \alpha \supset \alpha$

(A10) $\Box_S \alpha \equiv \neg \Diamond_S \neg \alpha$

(R1) From $\vdash \alpha$ and $\vdash \alpha \supset \beta$ infer $\vdash \beta$

We remark that the axiom (A1) and the inference rule (R1) come from propositional logic; hence the propositional subset of **ARL** is correctly handled. The other axioms capture the properties of the two families of modal operators. From an intuitive point of view, these axioms may be classified into two categories. The first one is concerned by the standards axioms (A2) – (A6) which capture the properties of double negation, commutativity, associativity, distributivity and DeMorgan’s laws. The specificity of the logic **ARL** lies in the second category. Axiom (A7) captures the conjunctive and disjunctive properties of operators \Box_S . Axiom (A8) is the key point of approximate reasoning. More precisely, if the parameter S is expanded by the atom p , then the system necessarily infers the tautology $p \vee \neg p$. Dually, if S is contracted by p , then the system can infer the antilogy $p \wedge \neg p$. Axiom (A9), often called T , demonstrates that reasoning under the scope of the modality \Box_S is sound. Finally, axiom (A10) captures the duality property between the modal operators \Box_S and \Diamond_S .

The following result gives soundness and completeness for the axiom system.

Theorem 2. *For every sentence α of the logic **ARL**,*

$$\vdash \alpha \text{ iff } \models \alpha.$$

Proof. The proof is presented in Appendix A.

3 Anytime deducers

After an excursion into the logic **ARL**, we now apply our results to the formal specifications of *anytime deducers*. In the context suggested by our approach, we define an anytime deducer as a knowledge based system that approximates the inference process by using an increasing sequence of resource parameters. Intuitively, the more time the system has to evaluate the query, the more resources it can spend. However, anytime deducers are *on-line reasoners*; there is no notion of directly processing the original knowledge base to obtain an approximation to it. This last requirement will be incorporated in the next section.

Following Levesque [11], we specify an anytime deducer as an “abstract type” that interacts with the user through a given set of service routines. To this end, we focus on two core operations, ASK and TELL, which allow a user to query the knowledge base and to add a new information to it. In the following definition, L_P denotes the set of all declarations of the logic **ARL**.

Definition 3. An *anytime deducer* consists of an operation TELL from $L_P \times L_P$ to L_P , and an operation ASK from $L_P \times 2^P \times L_P$ to $\{\text{YES}, \text{NO}, *\}$ respectively defined as follows:

$$\begin{aligned} \text{TELL}(A, \alpha) &= A \wedge \alpha, \\ \text{ASK}(A, S, \alpha) &= \begin{cases} \text{YES}, & \text{if } \models \Box_S (A \supset \alpha), \\ \text{NO}, & \text{if } \not\models \Diamond_S (A \supset \alpha), \\ *, & \text{otherwise.} \end{cases} \end{aligned}$$

The basic difference with the standard approach to knowledge representation relies on ASK operation that explicitly includes the notion of resource in order to capture the inference capabilities of the system.

Based on these considerations, the anytime deduction process is defined by an increasing sequence of resource parameters $\langle S_0 = \emptyset \cdots \subset S_i \cdots \subset S_n = P \rangle$ that approximate the set $\mathcal{A} = \{\alpha : \models A \supset \alpha\}$, by means of two dual families of sets $\mathcal{A}_i^\square = \{\alpha : \models \square_{S_i}(A \supset \alpha)\}$ and $\mathcal{A}_i^\diamond = \{\alpha : \models \diamond_{S_i}(A \supset \alpha)\}$. If we prove membership in any \mathcal{A}_i^\square then we have proved membership in \mathcal{A} . Dually, if we disprove membership in any \mathcal{A}_i^\diamond then we have disproved membership in \mathcal{A} . This stepwise process has the important advantage that the iteration may be stopped when a confirming answer is already obtained for a small index i . This yields a potentially drastic reduction of the computational costs. The following properties clarify the interest of our logic in the setting of anytime deduction.

Theorem 4 (Monotonicity). *For any declaration α and any resource parameters S and S' such that $S \subseteq S'$,*

- (1) if $\models \square_S \alpha$ then $\models \square_{S'} \alpha$,
- (2) if $\not\models \diamond_S \alpha$ then $\not\models \diamond_{S'} \alpha$.

Proof. Let us examine part (1). Assume that $\models \square_S \alpha$ and $\not\models \square_{S'} \alpha$. In the first case, for any possible world w and any valuation v such that $w \sim_S v$, we have $v \models_1 \alpha$. In the second case, there exists a possible world w' and a valuation v' such that $w' \sim_{S'} v'$ and $v' \not\models_1 \alpha$. Let us define a new valuation v'' such that $\forall p \in S, v''(p) = w'(p)$ and $\forall q \notin S, v''(q) = \{\}$. It is clear that $w \sim_S v''$. Moreover, we have $v'' \subseteq v'$. By application of lemma 1, it follows that $v'' \not\models_1 \alpha$. Therefore, we obtain $w \not\models_1 \square_S \alpha$, but this contradicts the hypothesis that $\models \square_S \alpha$. A dual argument applies to part (2).

Corollary 5 (Convergence). *For any declaration α ,*

- (1) if $\models \alpha$ then there exists a resource parameter S such that $\models \square_S \alpha$,
- (2) if $\not\models \alpha$ then there exists a resource parameter S such that $\not\models \diamond_S \alpha$.

Theorem 6 (Duality). *For any declaration α :*

- (1) $\models \square_S \alpha$ iff $\diamond_S \neg \alpha$ is unsatisfiable,
- (2) $\not\models \diamond_S \alpha$ iff $\square_S \neg \alpha$ is satisfiable.

Proof. Let us examine part (1). Assume that $\square_S \alpha$ is valid and that $\diamond_S \neg \alpha$ is satisfiable. By application of axioms (A10) and (A2), it follows that $\neg \square_S \alpha$ is satisfiable. Therefore, there exists a possible world w such that $w \models_1 \neg \square_S \alpha$. So, it follows that $w \models_0 \square_S \alpha$ and $w \not\models_1 \square_S \alpha$, but this contradicts the hypothesis that $\square_S \alpha$ is valid. Dual considerations hold for part (2).

Theorem 7 (Complexity). *For any declaration α and any resource parameter S , there exists an algorithm for deciding whether $\square_S \alpha$ is satisfiable and $\diamond_S \alpha$ is satisfiable which runs in $O(|\alpha| \cdot 2^{|S|})$.*

Proof. We begin to show that in a S -equivalence relation, at most one valuation of each equivalence class is needed for the satisfiability test. The sentence $\Box_S \alpha$ is satisfiable if there exists a possible world w such that $w \models_1 \Box_S \alpha$. Therefore, for any valuation v such that $v \sim_S w$, we have $v \models_1 \alpha$. Let us define the valuation v_\perp such that $\forall p \in S, v_\perp(p) = w(p)$ and $\forall q \notin S, v_\perp(q) = \{\}$. It is clear that $w \sim_S v_\perp$. By application of lemma 1, if $v_\perp \models_1 \alpha$ then $v \models_1 \alpha$ pour any v such that $w \sim_S v$. Therefore, $w \models_1 \Box_S \alpha$ iff $v_\perp \models_1 \alpha$. Now we turn to the satisfiability test. For each valuation v_\perp , the truth value $v_\perp \models_1 \alpha$ can be determined in $O(|\alpha|)$ time. Since there exists $2^{|S|}$ valuations v_\perp , checking whether $\Box_S \alpha$ is satisfiable can be done in $O(|\alpha| \cdot 2^{|S|})$ time. A dual argument applies to the sentence $\Diamond_S \alpha$.

Notice that the above complexity result is just the worst case upper bound of an enumeration algorithm. Although such an analysis is important, we do not claim the “brute force” method is the most feasible one. Actually, in the case of clausal theories, we can use a resolution based algorithm which computes the satisfiability of $\Box_{S_i} A$ and $\Diamond_{S_i} A$ by means of an increasing sequence of parameters S_i . For $S_0 = \emptyset$, this respectively corresponds to checking whether A is the empty base and A contains the empty clause. For a given theory A_i which is not empty and does not contain the empty clause, the procedure starts to resolve all clauses in A_i upon the literals p_{i+1} and $\neg p_{i+1}$, next eliminates all clauses in A_i containing these literals, and then checks whether the resulting theory is the empty base or contains the empty clause. It is interesting to remark that such an algorithm is indeed *incremental*; the procedure is able to exploit information gained in previous steps and does not require to perform all computations from scratch.

The correct choice of S is crucial for the usefulness of deduction. Taking to the extreme, when S is chosen incorrectly, anytime deduction may end up as expensive as classical deduction. From this perspective, several heuristics have been proposed in the literature [6,9,10,22]. For example, in a resolution based algorithm, the atoms of S may be dynamically chosen using the *minimal diversity heuristic* advocated in [6]. The diversity of an atom p is the product of the number of positive occurrences by the number of negative occurrences of p in the theory. This notion is based on the observation that an atom can be resolved upon only when it appears both positively and negatively in different clauses. Hence, choosing an atom p whose diversity is minimal will minimize the number of resolvents that can be generated upon p . This heuristic may be augmented with others strategies such as *boolean constraint propagation* and the *minimal width heuristic*. These considerations are illustrated in the following example.

Example 8. Suppose we are given $A = \{(\neg a \vee b \vee c), (a \vee b \vee \neg d), (a \vee \neg b \vee d), (\neg a \vee \neg b \vee c)\}$. We want to prove that A is satisfiable. Hence, we need to find a subset S of $\{a, b, c, d\}$ such that $\Box_S A$ is satisfiable. Starting with $S = \emptyset$ and using the minimal diversity heuristic, we gradually add the atoms b and d to S . This is sufficient for proving that A is satisfiable. Now, we want to show that $a \supset c$ is entailed by A . Hence, we need to find a subset S such that $\Diamond_S (A \wedge a \wedge \neg c)$ is unsatisfiable. Using boolean constraint propagation, we iteratively add to S the atoms a, c and b . This is sufficient for proving that $(A \wedge a \wedge \neg c)$ is unsatisfiable. Therefore, $a \supset c$ is indeed a logical consequence of A .

4 Anytime compilers

In this section, we extend the concepts developed so far to the formal specifications of *anytime compilers*. Such systems can perform *off-line reasoning*: they approximate the original knowledge base by allowing a sequence of more and more powerful data structures. The quality of compilation depends on the computational resources that have been spent. However, the computational cost of compilation is amortized over a potentially very large set of queries and the resulting data structures can be used in processing each query.

In the setting of knowledge compilation, it is well-known that every knowledge base has two specific normal forms, namely a conjunctive normal form (CNF) and a disjunctive normal form (DNF), from which queries can be efficiently answered. These normal forms are computed by means of the so-called *prime implicates* and *prime implicants*. An attractive property of this approach stems from the fact that the program used to generate the normal forms can be stopped before completion. More specifically, an interruptible process for generating prime implicates is sound but incomplete with respect to exact compilation. On the other hand, an interruptible prime implicant generation process is unsound but complete with respect to exact compilation. Based on these considerations, we present a method for generating prime implicates and prime implicants defined in terms of the logic **ARL**.

To that end, we introduce some additional definitions. A *literal* is an atom or its negation, a *clause* is a finite conjunction of literals and a *term* is a finite disjunction of literals. When there is no risk of confusion, we shall model clauses and terms as sets of literals. A clause γ is called a *S-implicate* of a knowledge base A , if $\models \Box_S (A \supset \gamma)$ and γ does not contain two complementary literals. Dually, a term τ is a *S-implicant* of a A , if $\models \Box_S (\tau \supset A)$ and τ does not contain two complementary literals. A clause γ is called a *prime S-implicate* of A , if γ is a *S-implicate* of A and for every other *S-implicate* γ' of A , we have $\gamma' \not\subset \gamma$. In a similar way, a term τ is called a *prime S-implicant* of A , if τ is a *S-implicant* of A , and for every other *S-implicant* τ' of A , we have $\tau' \not\subset \tau$.

In the remaining paper, the conjunction of all the prime *S-implicates* of a knowledge base A is denoted $\text{PIC}(A, S)$ and the disjunction of all the prime *S-implicants* of A is denoted $\text{PID}(A, S)$. When clear from the context, such sentences will be respectively modeled as sets of clauses and sets of terms. Now, we have the formal tools for specifying anytime compilers.

Definition 9. An *anytime compiler* consists of an operation **TELL** from $L_P \times 2^P \times L_P$ to $L_P \times L_P$, and an operation **ASK** from $L_P \times L_P \times L_P$ to $\{\text{YES}, \text{NO}, *\}$ respectively defined as follows:

$$\begin{aligned} \text{TELL}(A, S, \alpha) &= \left(\text{PIC}(A \wedge \alpha, S), \text{PID}(A \wedge \alpha, S) \right), \\ \text{ASK}(A_{\text{PIC}}, A_{\text{PID}}, \alpha) &= \begin{cases} \text{YES}, & \text{if } \models A_{\text{PIC}} \supset \alpha, \\ \text{NO}, & \text{if } \not\models A_{\text{PID}} \supset \alpha, \\ *, & \text{otherwise.} \end{cases} \end{aligned}$$

The basic idea underlying the above definition is to invoke compilation during the TELL operation. Hence, the resulting data structures A_{PIC} and A_{PID} may be used in the ASK operation in order to answer many queries.

As for deduction, the compilation process may be modeled by an increasing sequence of parameters $\langle S_0 = \emptyset \cdots \subset S_i \cdots \subset S_n = P \rangle$ that approximate the deductive closure of A , denoted \mathcal{A} , by means of two dual families of sets $\mathcal{A}_i^{\text{PIC}} = \{\alpha : \models \text{PIC}(A, S) \supset \alpha\}$ and $\mathcal{A}_i^{\text{PID}} = \{\alpha : \models \text{PID}(A, S) \supset \alpha\}$. For a given index i , if we prove membership in $\mathcal{A}_i^{\text{PIC}}$, then we have also proved membership in \mathcal{A} . On the other hand, if we disprove membership in $\mathcal{A}_i^{\text{PID}}$, then we have also disproved membership in \mathcal{A} . The reader might wonder at this point if there exists a close relationship between the two previous families \mathcal{A}_i^{\square} and \mathcal{A}_i^{\diamond} generated during anytime deduction, and the two families $\mathcal{A}_i^{\text{PIC}}$ and $\mathcal{A}_i^{\text{PID}}$ defined during anytime compilation. In fact, as stated in the following theorem, there is a one to one correspondence between these families.

Theorem 10 (Correspondence). *For any knowledge base A , any declaration α , and any resource parameter S ,*

- (1) $\models \text{PIC}(A, S) \supset \alpha$ iff $\models \square_S (A \supset \alpha)$,
- (2) $\models \text{PID}(A, S) \supset \alpha$ iff $\models \diamond_S (A \supset \alpha)$.

Proof. We begin to introduce some useful definitions. We denote \mathcal{V}_S^\top the set of valuations v such that for every atom $p \in P$, $v(p) = \{0\}$ or $v(p) = \{1\}$ if $p \in S$, and $v(p) = \{0, 1\}$ otherwise. Dually, \mathcal{V}_S^\perp denotes the set of valuations v such that for every $p \in P$, $v(p) = \{0\}$ or $v(p) = \{1\}$ if $p \in S$, and $v(p) = \{\}$ otherwise.

Let us examine part (1). A sufficient condition for proving (1) is to state that for any $v \in \mathcal{V}_S^\top$, $v \models_1 A$ iff $v \models_1 \text{PIC}(A, S)$.

- Suppose that there is a $v \in \mathcal{V}_S^\top$ such that $v \models_1 A$ and $v \not\models_1 \text{PIC}(A, S)$. In this case, there must exist at least one clause γ in $\text{PIC}(A, S)$ such that $v \not\models_1 \gamma$. Since γ is a prime S -implicate, the sentence $\diamond_S (A \wedge \neg\gamma)$ is unsatisfiable. So, either $v \not\models_1 \neg\gamma$ holds or $v \not\models_1 A$ holds. In the first case, we would obtain $v \not\models_1 \gamma \vee \neg\gamma$, but this is impossible since from definition of v there must exist at least one possible world $w \subseteq v$ such that $w \models_1 \gamma \vee \neg\gamma$ and by lemma 1, $v \models_1 \gamma \vee \neg\gamma$. So, $v \not\models_1 A$ holds, hence contradiction.
- Now, assume that there is a $v \in \mathcal{V}_S^\top$ such that $v \not\models_1 A$ and $v \models_1 \text{PIC}(A, S)$. Thus, for every prime S -implicate γ in $\text{PIC}(A, S)$, we must have $v \models_1 \gamma$. Suppose that A is unsatisfiable. Then it is easy to prove that $\text{PIC}(A, S)$ is either empty or contains the empty clause. In both cases, it follows that $v \not\models_1 \text{PIC}(A, S)$, hence contradiction. Now, suppose that A is satisfiable. Since γ is a prime S -implicate of A , then by theorem 4, it follows that γ is a prime implicate of A . Therefore, for every possible world w such that $w \models_1 A$ there exists at least one literal $l \in \gamma$ such that $w \models_1 l$. If for every $l \in \gamma$, we have $v \models_1 l$, then there exists at least one world $w \subseteq v$ such that $w \models_1 l$ and $w \models_1 A$. By lemma 1, it follows that $v \models_1 A$, hence contradiction. If there exists a $\gamma' \subset \gamma$ such that $v \not\models_1 \gamma'$, then for every possible world w such that $w \models_1 A$, we have $w \models_1 \gamma'$. Since $\diamond_S (A \wedge \neg\gamma')$ is unsatisfiable, then γ' is a S -implicate of A . Therefore $\gamma \notin \text{PIC}(A, S)$, hence contradiction.

We now turn to part (2). In a dual way, a sufficient condition for proving (2) is to show that for any $v \in \mathcal{V}_S^\perp$, $v \models_1 A$ iff $v \models_1 \text{PID}(A, S)$.

- Suppose that there exists a $v \in \mathcal{V}_S^\perp$ such that $v \models_1 A$ and $v \not\models_1 \text{PID}(A, S)$. If $v \models_1 A$, then viewing v as a set of literals, we must have $\models v \supset A$. Moreover, from definition of v , it follows that $\models \Box_S (v \supset A)$. Hence, v is a S -implicant of A . It is clear that v cannot be a prime S -implicant, because otherwise we would have $v \models_1 \text{PID}(A, S)$. Therefore, there exists a term $\tau \in \text{PID}(A, S)$ such that $\tau \subset v$ and $\tau \not\models_1 A$. However, by lemma 1, it follows that $v \not\models_1 A$, hence contradiction.
- Now, suppose that there exists a $v \in \mathcal{V}_S^\perp$ such that $v \not\models_1 A$ and $v \models_1 \text{PID}(A, S)$. In this case, there exists a term $\tau \in \text{PID}(A, S)$ such that $v \models_1 \tau$, that is, $\tau \subseteq v$. Since $\Box_S (\tau \supset A)$ is valid, we must also have $\tau \models_1 A$. By lemma 1 it follows that $v \models_1 A$, hence contradiction.

The correspondence result above is very interesting because most of the properties stated for anytime deduction also hold in the setting of anytime compilation. As an example, for any parameters S and S' such that $S \subseteq S'$, we can state that $\text{PIC}(A, S')$ is as complete as $\text{PIC}(A, S)$ and that $\text{PID}(A, S')$ is as sound as $\text{PID}(A, S)$. The next result is even stronger than monotonicity; we show that anytime compilation is *incremental*, using information gained in previous steps.

Theorem 11 (Incrementality). *For any knowledge base A , and any resource parameters S and S' such that $S \subseteq S'$,*

- (1) $\text{PIC}(A, S) \subseteq \text{PIC}(A, S')$,
- (2) $\text{PID}(A, S) \subseteq \text{PID}(A, S')$.

Proof. Let us demonstrate part (1). Suppose that $\gamma \in \text{PIC}(A, S)$ and $\gamma \notin \text{PIC}(A, S')$. Since $\models \Box_S (A \supset \gamma)$ holds, then by theorem 4, $\models \Box_{S'} (A \supset \gamma)$ also holds. Hence, there must exist a clause γ' such that $\gamma' \subset \gamma$ and $\models \Box_{S'} (A \supset \gamma')$. However, in this case it is clear that $\models \Box_S (A \supset \gamma')$ holds. So $\gamma \notin \text{PIC}(A, S)$, hence contradiction. An analogous strategy applies to part (2).

The complexity result presented below clarifies the interest of the compilation process from a computational point of view. More precisely, we demonstrate that entailment of CNF queries can be computed in time polynomial of the size of resulting data structure plus the size of the query.

Theorem 12 (Complexity). *For any knowledge base A , any resource parameter S and any declaration α in CNF, there is an algorithm for deciding whether $\text{PIC}(A, S) \supset \alpha$ is valid (resp. $\text{PID}(A, S) \supset \alpha$ is valid) which runs in $O(|\text{PIC}(A, S)| + |\alpha|)$ (resp. $O(|\text{PID}(A, S)| + |\alpha|)$).*

Proof. $\models \text{PIC}(A, S) \supset \alpha$ holds iff every non tautological clause γ' of α there is a prime S -implicate γ of $\text{PIC}(A, S)$ such that $\gamma \subseteq \gamma'$. This can be done in $O(|\text{PIC}(A, S)| + |\alpha|)$. On the other hand, $\models \text{PID}(A, S) \supset \alpha$ holds iff every clause γ has a non-empty intersection with every S -implicant τ of $\text{PID}(A, S)$. This can be done in $O(|\text{PID}(A, S)| + |\alpha|)$.

Clearly, the effectiveness of compilation for subsequent query processing depends on the size of its resulting data structures. In the setting suggested by our approach, a knowledge base can have at most $3^{|S|}$ S -implicates and $3^{|S|}$ S -implicants. Using the results by Chandra and Markowsky in [4], the same knowledge base may have on average $3^{|S|}/|S|$ prime S -implicates and $3^{|S|}/\sqrt{|S|}$ prime S -implicants. From this point of view, the interest of anytime compilation is that it has potential to greatly decrease the inconvenience of using exact compilation: since off-line reasoning may be too space demanding, it is clearly desirable to be able to process queries before completion.

Several algorithms can be used to compute prime S -implicates and prime S -implicants. In the first case, we may conceive a stepwise procedure which starts by computing $\text{PIC}(A, S)$ for $S = \emptyset$ and that iteratively increases the value of S . For $S = \emptyset$, this corresponds to checking whether the knowledge base A contains the empty clause or not. For a theory A which does not contain the empty clause, we check each possible implicate γ in turn by adding its negation to the base and then testing $\diamond_S (A \wedge \neg\gamma)$ for satisfiability. If the sentence is refuted, then γ is a S -implicate of A . By allowing an increasing sequence of parameters S , we can notice already subsumed implicates and not count these in prime S -implicate generation. As far as $\text{PID}(A, S)$ is concerned, dual considerations hold.

As for anytime deduction, the correct choice of the parameter S is important for the usefulness of anytime compilation. This choice may be heuristic; in this case, the atoms of S are iteratively selected to minimize the predicted number of generated prime S -implicates and prime S -implicants, using strategies suggested in [6,18,19]. Alternatively, the choice of S may be guided by query answering considerations. The letters selected during deduction are used in turn for compilation. In other words, the work done for one query is saved for use in answering the next query. These considerations are illustrated in the following example.

Example 13. We are given $A = \{(a \vee b \vee c), (a \vee b \vee \neg c), (b \vee d \vee \neg e), (b \vee \neg d \vee e)\}$. Suppose, we want to generate at least one prime S -implicate of A . Starting with $S = \emptyset$ and using the minimal diversity heuristic, we gradually add to S the atoms b , a and c . This is sufficient to obtain $\text{PIC}(A, S) = \{(a \vee b)\}$. Now, suppose that the system is frequently asked CNF queries containing the atom b . In this case, we iteratively add to S the atoms b , d and e . Hence, we obtain $\text{PIC}(A, S) = \{(b)\}$. Notice that in both scenarios we have $\text{PID}(A, S) = \{(b)\}$.

5 Conclusion

In this paper, we have dealt with the problem of reasoning in propositional knowledge bases focusing on two attractive approaches, namely, anytime deduction and anytime compilation. The first one offers a compromise between the time complexity needed to compute approximate answers and the quality of these answers. The second one proposes a trade-off between the space complexity of the compiled knowledge base and the number of possible answers that can be efficiently processed by the resulting data structure. Our aim was to provide a unifying, logic oriented framework which handles these two approaches

and that enables us to specify anytime reasoners. We have stressed on a sound and complete multi-modal logic, named **ARL**, which generalizes and expands in several directions previous methods concerning approximate deduction [2,5,17] and anytime compilation [7,15,18]. Based on this logic, we have illustrated that the framework integrates several major features: resource-bounded reasoning, improvability, dual reasoning and off-line processing.

We believe that the results reported here are interesting and worth of further investigations. We outline some of them. A first extension is concerned by the empirical analysis of the parameter S . In particular, it has been found in [21] that random “3-SAT” knowledge bases can be classified in three categories, namely *under-constrained*, *over-constrained* and *critically-constrained*, according to the ratio clauses-to-atoms. An important issue here is to examine the relationship between this ratio and the resource parameter S . This will give an estimation of the number of computational resources (i.e. atoms) required to perform deduction and compilation tasks in each category of knowledge bases. A second extension is to study anytime reasoning in the setting of *pseudo-first-order logic*, which has received a great deal of interest in database theory (e.g. [24]). These representation languages are defined from a finite domain of discourse without function symbols. However, although every first order knowledge base can be replaced by an equivalent propositional theory, the size of the theory may be exponentially larger than the initial base. Hence, formal extensions of our logic should be done in order to control such a source of complexity. A third possible extension is to consider anytime reasoning in a nonmonotonic setting. As an example, in a multi-agent system, a reasoner is often confronted with uncertain and inconsistent information [9,10]. In such circumstances, it is necessary to incorporate conflict resolution methods and preference orderings which involve additional sources of complexity. Important issues such as *anytime nonmonotonic reasoning* and *anytime recompilation* should play a key role in this setting.

References

1. N. D. Belnap. A useful four-valued logic. In J. M. Dunn and G. Epstein, editors, *Modern Uses of Multiple-Valued Logic*, pages 8–37. Reidel, Dordrecht, 1977.
2. M. Cadoli. *Tractable reasoning in artificial intelligence*, volume 941 of *Lecture Notes in Artificial Intelligence*. Springer-Verlag Inc., New York, NY, USA, 1995.
3. M. Cadoli and M. Schaerf. Approximate inference in default reasoning and circumscription. In B. Neumann, editor, *Proceedings 10th European Conference on Artificial Intelligence*, pages 319–323, Vienna, Austria, 1992. John Wiley & Sons.
4. A. Chandra and G. Markowsky. On the number of prime implicants. *Discrete Mathematics*, 24:7–11, 1978.
5. M. Dalal. Semantics of an anytime family of reasoners. In W. Wahlster, editor, *Proceedings of the 12th European Conference on Artificial Intelligence (ECAI-96)*, pages 360–364. Wiley & Sons, 1996.
6. R. Dechter and I. Rish. Directional resolution: The davis-putnam procedure, revisited. In P. Torasso, J. Doyle, and E. Sandewall, editors, *Proceedings of the 4th International Conference on Principles of Knowledge Representation and Reasoning*, pages 134–145, Bonn, Germany, 1994. Morgan Kaufmann.

7. A. del Val. Tractable databases: How to make propositional unit resolution complete through compilation. In J. Doyle, E. Sandewall, and P. Torasso, editors, *Proceedings of the 4th International Conference on Principles of Knowledge Representation and Reasoning*, pages 551–561, Bonn, 1994. Morgan Kaufmann.
8. J. M. Dunn. Intuitive semantics for first-degree entailments and coupled trees. *Philosophical Studies*, 29:149–168, 1976.
9. F. Koriche. Approximate reasoning about combined knowledge. In M. P. Singh, A. S. Rao, and M. J. Wooldridge, editors, *Intelligent Agents Volume IV*, volume 1365 of *Lecture Notes in Artificial Intelligence*, pages 259–273. Springer Verlag, Berlin, Germany, 1998.
10. F. Koriche. *Approximate Reasoning in Cooperating Knowledge Based Systems*. PhD thesis, Université de Montpellier II, Montpellier, France, January 1998.
11. H. J. Levesque. Foundations of a Functional Approach to Knowledge Representation. *Artificial Intelligence*, 23:125–212, 1984.
12. H. J. Levesque. A logic of implicit and explicit belief. In R. J. Brachman, editor, *Proceedings of the 6th National Conference on Artificial Intelligence*, pages 198–202, Austin, Texas, 1984. William Kaufmann.
13. P. Marquis. Knowledge compilation using theory prime implicates. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, volume 2, pages 837–843, Toronto, Canada, 1995. Morgan Kaufmann.
14. J. McCarthy. Programs with common sense. In *Proceedings of the Symposium on Mechanisation of Thought Processes*, volume 1, pages 77–84, London, 1958. Her Majesty's Stationery Office.
15. T. Ngair. A new algorithm for incremental prime implicate generation. In *Proceedings 13th International Joint Conference on Artificial Intelligence*, volume 1, pages 46–51, Chambéry, France, aug 1993. Morgan Kaufmann.
16. R. Reiter and J. de Kleer. Foundations of assumption-based truth maintenance systems: Preliminary report. In K. Forbus and H. Shrobe, editors, *Proceedings of the 6th National Conference on Artificial Intelligence*, pages 183–189, Seattle, (WA) USA, July 1987. Morgan Kaufmann.
17. M. Schaerf and M. Cadoli. Tractable reasoning via approximation. *Artificial Intelligence*, 74:249–310, 1995.
18. R. Schrag. Compilation for critically constrained knowledge bases. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence and the Eighth Innovative Applications of Artificial Intelligence Conference*, pages 510–516, Menlo Park (CA) USA, August 1996. AAAI Press / MIT Press.
19. R. Schrag and J. Crawford. Implicates and prime implicates in random 3-SAT. *Artificial Intelligence*, 81(1-2):199–222, 1996.
20. B. Selman and H. Kautz. Knowledge compilation and theory approximation. *Journal of the ACM*, 43(2):193–224, 1996.
21. B. Selman, D. Mitchell, and H. Levesque. Generating hard satisfiability problems. *Artificial Intelligence*, 81:17–30, 1996.
22. A. ten Teije and F. van Harmelen. Computing approximate diagnoses by using approximate entailment. In L. C. Aiello, J. Doyle, and S. Shapiro, editors, *Proceedings of the Fifth International Conference on Principles of Knowledge Representation and Reasoning*, pages 256–267, San Francisco CA, USA, 1996. Morgan Kaufmann.
23. P. Tison. Generalized consensus theory and application to the minimization of boolean functions. *IEEE Transactions on electronic computers*, 4:446–456, 1967.
24. M. Y. Vardi. Querying logical databases. In *Proceedings of the Third ACM SIGACT-SIGMOD Symposium on Principles of Database Systems*, pages 57–65, Waterloo, Ontario, Canada, 1985.

Appendix A. Proof of soundness and completeness

The following results give soundness and completeness for the axiom system.

Theorem 14 (Soundness). *For every sentence α of the logic **ARL**,*

$$\text{if } \vdash \alpha \text{ then } \models \alpha.$$

Proof. It is easy to see that axioms (A1)–(A6) are sound, and that the inference rule (R1) preserves validity. Let us examine the other axioms.

- Axiom (A7). The only nontrivial case is the soundness of $\Box_S(\alpha \vee \beta) \supset \Box_S\alpha \vee \Box_S\beta$. Suppose that this sentence is not valid. Then there exists a possible world w such that $w \models_1 \Box_S(\alpha \vee \beta)$ and $w \not\models_1 \Box_S\alpha \vee \Box_S\beta$. In the first case, for every valuation v such that $v \sim_S w$, we have $v \models_1 \alpha \vee \beta$. In the second case, there exists two valuations v', v'' such that $v' \sim_S w$, $v'' \sim_S w$, $v' \not\models_1 \alpha$, and $v'' \not\models_1 \beta$. Let us define a new valuation u , such that $u(p) = w(p)$ for every $p \in S$, and $u(q) = \{\}$ for every $q \notin S$. It is clear that w, v, v', v'' and u belong to the same equivalence class defined for \sim_S . Moreover, $u \subseteq v'$ and $u \subseteq v''$. By lemma 1, we obtain $u \not\models_1 \alpha$ and $u \not\models_1 \beta$. So $u \not\models_1 \alpha \vee \beta$ and therefore $w \not\models_1 \Box_S(\alpha \vee \beta)$, hence contradiction.
- Axiom (A8). Let us examine the first sentence. Suppose that $\Box_{S \cup \{p\}}(p \vee \neg p)$ is not valid. Then there must exist a possible world w such that $w \not\models_1 \Box_{S \cup \{p\}}(p \vee \neg p)$. In this case, there exists a valuation v such that $w \sim_{S \cup \{p\}} v$ and $v \not\models_1 p \vee \neg p$. However, since w is a possible world, we have $1 \in w(p)$ or $0 \in w(p)$. Moreover, since $p \in (S \cup \{p\})$ it follows that $1 \in v(p)$ or $0 \in v(p)$. So $v \models_1 p \vee \neg p$, hence contradiction. We now turn to the second sentence. $\Diamond_{S - \{p\}}(p \wedge \neg p)$ is valid iff for every possible world w there exists a valuation v such that $w \sim_{S - \{p\}} v$ and $v \models_1 p \wedge \neg p$. Let us define a new valuation v' such that $v'(p) = \{0, 1\}$ and for every $q \in (S - \{p\})$, $v'(q) = w(q)$. It is clear that $v' \models_1 p \wedge \neg p$. Moreover, for every possible world w , we have $w \sim_{S - \{p\}} v'$. Therefore, $\Diamond_{S - \{p\}}(p \wedge \neg p)$ is valid.
- Axiom (A9). Suppose that $\Box_S\alpha \supset \alpha$ is not valid. Then there exists a possible world w such that $w \models_1 \Box_S\alpha$ and $w \not\models_1 \alpha$. If $w \models_1 \Box_S\alpha$ then for every valuation v such that $w \sim_S v$, we have $v \models_1 \alpha$. Since \sim_S is reflexive, it follows that $w \models_1 \alpha$, hence contradiction.
- Axiom (A10). Let us examine the first implication (if). Suppose that $\Box_S\alpha \supset \neg\Diamond_S\neg\alpha$ is not valid. Then there exists a possible world w such that $w \models_1 \Box_S\alpha$ and $w \not\models_1 \neg\Diamond_S\neg\alpha$. In the first case, for any valuation v such that $v \sim_S w$, we have $v \models_1 \alpha$, while in the second case there exists a valuation v' such that $v' \sim_S w$ and $v' \models_0 \alpha$. It follows that $v' \models_1 \alpha \wedge \neg\alpha$. Let us define a new valuation v'' such that for every $p \in S$, $v''(p) = v'(p)$ if $p \in S$, and for every $q \notin S$, if $v'(q) = \{\}$ then $v''(q) = \{0, 1\}$ and if $v'(q) = \{0, 1\}$ then $v''(q) = \{\}$. It is easy to show by induction on the structure of α that if $v' \models_1 \alpha \wedge \neg\alpha$ then $v'' \not\models_1 \alpha \vee \neg\alpha$. Moreover, it is clear that $v' \sim_S v''$. Therefore, it follows that $w \not\models_1 \Box_S\alpha$, hence contradiction. An analogous strategy applies to the second implication (only if).

Theorem 15 (Completeness). *For every sentence α of the logic **ARL**,*

$$\text{if } \models \alpha \text{ then } \vdash \alpha.$$

Proof. We divide the proof into three steps. In the first step, we introduce the notion of *maximal consistent set* and we examine several properties of these structures. In the second step, we show that every sentence of the language has two important normal forms, namely an *extended conjunctive normal form* (ECNF) and an *extended disjunctive normal form* (EDNF) which are useful for the satisfiability test. Finally, in the third step, we prove that every sentence in a maximal consistent set is satisfiable. We conclude the proof by showing that this result is sufficient to state completeness.

We start the first step by giving some definitions. A sentence α is *consistent* if its negation is not theorem (i.e. $\not\vdash \neg\alpha$). A finite set of sentences is consistent exactly if the conjunction of its sentences is consistent, and an infinite set of sentences is consistent exactly if all of its finite subsets are consistent. A sentence or a set of sentences is said to be *inconsistent* exactly if it is not consistent. Finally, a set A of sentences is called *maximal consistent* if it is consistent and for any sentence α of **ARL**, if $\alpha \notin A$ then $A \cup \{\alpha\}$ is inconsistent.

Lemma 16. *In the logic **ARL**, Every consistent set of sentences can be extended to a maximal consistent set of sentences. In addition, if A is a maximal consistent set, then it satisfies the following properties:*

- (1) *for every sentence α , exactly one of α and $\neg\alpha$ is in A ,*
- (2) *$\alpha \wedge \beta \in A$ iff $\alpha \in A$ and $\beta \in A$,*
- (3) *$\alpha \vee \beta \in A$ iff $\alpha \in A$ or $\beta \in A$,*
- (4) *if $\vdash \alpha$ then $\alpha \in A$,*
- (5) *if $\alpha \in A$ and $\vdash \alpha \supset \beta$ then $\beta \in A$.*

Proof. Straightforward, by using standard techniques of propositional logic.

We now turn to the second step. We first recall that a *literal* is an atom or its negation, a *clause* is a finite conjunction of literals and a *term* is a finite disjunction of literals. A declaration in *conjunctive normal form* (CNF) is a conjunction of clauses and a declaration in *disjunctive normal form* (DNF) is a disjunction of terms. A sentence α is said to be in *extended conjunctive normal form* (ECNF) if, treating subformulas of the form $\Box_S \beta$ and $\Diamond_S \beta$ as atoms, α is in CNF, and for each subformula $\Box_S \beta$ and $\Diamond_S \beta$ of α , β is in CNF. The notion of *extended disjunctive normal form* (EDNF) is defined in a dual way.

Lemma 17. *For every sentence α of the logic **ARL**,*

$$\vdash \alpha \equiv \alpha_{\text{ECNF}} \quad \text{and} \quad \vdash \alpha \equiv \alpha_{\text{EDNF}}.$$

Proof. By repeated applications of axioms (A2)-(A6) and rule (R1).

Now we have proved the two preparatory lemmas, we turn to the third step. We demonstrate that every sentence in a maximal consistent set is satisfiable in the semantics of our logic. This property may be formalized as follows.

Lemma 18. *If A is maximal consistent set of sentences of **ARL**, then there exists a possible world $w_A \in \mathcal{W}_P$ such that:*

$$\forall \alpha \in \mathbf{ARL}, \alpha \in A \text{ iff } w_A \models_1 \alpha.$$

Proof. Let w_A be defined in the following way: for every atom $p \in P$, $w_A \models_1 p$ iff $p \in A$. We prove the lemma by induction on the structure of α . If α is an atom p , this is immediate from the definition of w_A . The cases where α is a negation, a conjunction or a disjunction follow easily from parts (1), (2) and (3) of lemma 16. In the proof below, we concentrate on the case where α is of the form $\Box_S \beta$. The final case where α is of the form $\Diamond_S \beta$ directly follows from axiom (A10).

- (if direction). Suppose that $\Box_S \beta \in A$ and that $w_A \not\models_1 \Box_S \beta$. From the first assumption and by lemma 17, we have $\Box_S \beta_{\text{ECNF}} \in A$. So, by axiom (A7) and part (2) of lemma 16, for every clause γ of β_{ECNF} , we must have $\Box_S \gamma \in A$. Moreover, by axiom (A7) and part (3) of lemma 16, there exists at least one literal l of γ such that $\Box_S l \in A$. From the second assumption, we obtain $w_A \not\models_1 \Box_S \beta_{\text{ECNF}}$. Therefore, there must exist a clause γ' of β_{ECNF} , such that for every $l' \in \gamma'$, we have $w_A \not\models_1 \Box_S l'$. If $l' \in S$, then $w_A \not\models_1 l'$ and by the induction hypothesis we have $l' \notin A$. By axiom (A9) we obtain $\Box_S l' \notin A$, hence contradiction. If $l' \notin S$, by axiom (A8) we obtain $\Diamond_S (l' \wedge \neg l') \in A$, and by axiom (A10) we have $\neg \Box_S \neg (l' \wedge \neg l') \in A$. By application of axioms (A2),(A3) and (A6), it follows that $\neg \Box_S (l' \vee \neg l') \in A$. From part (1) of lemma 16 we first obtain $\Box_S (l' \vee \neg l') \notin A$ and by axiom (A7) it follows that $\Box_S l' \vee \Box_S \neg l' \notin A$. Finally, from part (3) of lemma 16, we obtain $\Box_S l' \notin A$, hence contradiction.
- (only if direction). Suppose that $w_A \models_1 \Box_S \beta$ and that $\Box_S \beta \notin A$. From the first assumption, we obtain $w_A \models_1 \Box_S \beta_{\text{ECNF}}$. So, for every clause γ of β_{ECNF} , there exists a literal l such that its atom is an element of S and $w_A \models_1 l$. However, by axiom (A8) we have $\Box_S (l \vee \neg l) \in A$. By axiom (A7), it follows that $\Box_S l \vee \Box_S \neg l \in A$, and from part (3) of lemma 17 we have $\Box_S l \in A$ or $\Box_S \neg l \in A$. Since $w_A \models_1 l$, by the induction hypothesis we have $l \in A$. By axiom (A9) we must obtain $\Box_S \neg l \notin A$, because otherwise we would have $\neg l \in A$. Therefore we have $\Box_S l \in A$, and by axiom (A7) it follows that $\Box_S \gamma \in A$. A similar method applies to each clause of β_{ECNF} . Therefore, by axiom (A7), it follows that $\Box_S \beta_{\text{ECNF}} \in A$. Finally, by lemma 17 we obtain $\Box_S \beta \in A$, hence contradiction.

Now, we have all the results in hand for concluding the proof. By lemma 16, we know that if a sentence is consistent, then it is a member of a maximal consistent set. But in lemma 18, we have shown that if a sentence belongs to such a set, then it is satisfiable. So, we have proved that every consistent sentence is satisfiable. This is sufficient to conclude that every valid sentence is provable.