Pablo F. Castro

Tableau Systems for Deontic Action Logics based on Finite Boolean Algebras, and Their Complexity

Abstract. We introduce a family of tableau calculi for deontic action logics based on finite boolean algebras (or DAL for short), these logics provide deontic operators (e.g., obligation, permission, prohibition) which are applied to a finite number of actions (the vocabulary of the logic); furthermore, in these formalisms, actions can be combined by means of boolean operators, this provides an expressive algebra of actions. We define a tableau calculus for the basic logic and then we extend this calculus to cope with extant variations of this formalism; we prove the soundness and completeness of these proof systems. In addition, we investigate the computational complexity of the satisfiability problem for DAL and its extensions; we show this problem is NP-Complete when the number of actions considered is fixed, and it is Σ_2^p-Hard (in Stockmeyer's polynomial hierarchy) when the number of actions is taken as an extra parameter. The tableau systems introduced here can be implemented in PSPACE, this seems reasonable taking into consideration the computational complexity of the logics.

Keywords: Deontic Action Logics, Tableau Systems, Computation Complexity, SAT

1. Introduction

Deontic logic is a branch of logic and philosophy that is devoted to the study of the logical properties of norms and related notions, one of the main goals in this field is the formal understanding of normative concepts such as *permission*, *obligation* and *prohibition*. The origin of the logical analysis of norms, and related concepts, can be traced back to Aristotle and the scholastic philosophers, a detailed discussion about the history of deontic logic is given in [17, 13, 1]. It is worth noting that the first authors in proposing formal deontic systems were Mally [18], von Wright [28] and Kalinowski [15]; since then philosophers, logicians and computer scientists have proposed different formalisms to reason about normative systems. Interestingly, in the last few decades, deontic logics have been found particularly useful for reasoning about computing systems and for performing automated analysis of legal systems, a good account of this is given in [29].

Presented by ; *Received*

Among the different deontic logics proposed in the literature, we can distinguish between two kinds of systems. On the one hand, we have the *ought-to-be* deontic systems in which deontic operators are applied to states of affairs, a simple example of this approach is given by the sentence: *the fence ought to be white*, note that this statement prescribes a state of the fence, or more generally it is a normative statement about a predicate describing a scenario or situation. *standard deontic logic* (or SDL) [8] has been one of the most popular ought-to-be systems, this logic has sparked interesting discussions about formal deontic logics, in particular regarding the so-called "paradoxes"; a paradox in deontic logic is a normative statement whose truth value contradicts our intuition about norms, a good account of paradoxes in SDL is given in [20]. Ought-to-be logics have been deeply investigated by the community of deontic logic, standard references are [1, 8] and we refer the interested reader to them. On the other hand, other authors have introduced the so-called *ought-to-do* systems (the ones we are interested in here) in which normative statements are used for prescribing actions and their consequences or, as is called in the literature, *practitions*. As a simple example of this approach consider the following sentence: *it is allowed to drive while smoking*, note that in this case the range of the prescription is a composed action (the action composed of driving and smoking at the same time). The importance of deontic action logics in computer science has been pointed out by several authors, e.g. [29]. Actions can be identified with basic programs, and action combinators can be think of as being program operations: action conjunction is identified with parallel execution of programs, action disjunction with nondeterministic choice, and action complement with the execution of alternative code (many other operators can be considered). These kinds of logics have been shown to be useful for reasoning about legal systems, fault-tolerant systems and electronic contracts, just to name a few examples.

Over the last decades, several logics based on these ideas have been proposed in the literature, *dynamic deontic logic* (DDL for short) [19] is one of the most popular ought-to-do systems, roughly speaking, this formalism reduces deontic operators to standard constructions of dynamic logic [12]. Recall that formulas of dynamic logic have the form $[\alpha]\varphi$, where α is an action and φ is a formula, this formula can be read as: *after executing action α, φ is necessarily true.* In this setting, for example, prohibition is formalized as: $\mathbf{F}(\alpha) \equiv [\alpha]V$, here V is a fresh propositional symbol indicating a violation, intuitively, this formula asserts that action α is forbidden iff every execution of α leads to a violation. The main benefit of this approach, as explained in [19], is the possibility of getting rid of usual problems of ought-to-be deontic logics. For instance, in DDL, deontic operators cannot be nested, avoiding in this way complex formulas that do not have a clear semantics; also the notion of state change appears explicitly in formulas (in contrast to standard deontic logic) which helps to avoid some paradoxes of SDL, the interested reader is referred to [19] for examples about this. However, as detailed in [2], dynamic deontic logic has its own problems and paradoxes. One of the drawbacks of dynamic deontic logic is the strong interplay between standard modalities and deontic operators, for instance, the following formula is valid in DDL: $\mathbf{P}(\alpha) \rightarrow \langle\alpha\rangle\top$, whose intuitive reading is: *allowed actions can be executed*, it is easy to devise scenarios where this is not true, suppose a person that must pay taxes, the person is allowed to pay the taxes, but perhaps she does not have money to do so. As we describe below, many authors have pointed out that deontic operators should be defined without resorting to standard modalities, we review these approaches later in this section.

Several formalisms based on DDL have been proposed in the literature; in [4], Broersen describes another possible formulation of dynamic deontic logic where different violation constants are used for defining obligation and permission, avoiding in this way the interdefinibility of these operators as given by Meyer. Also, Broersen uses a different version of action complement in his logic, arguing that this approach is more appropriate for computer science. A modal action system (called MAL) similar to dynamic deontic logic is investigated in [16], a similar relation to that imposed by Meyer between permission and state properties is introduced in the semantics. In this work a partial proof system is presented, although its properties are not investigated in detail.

Other authors have proposed what are called *process oriented norms*, that is, prescriptions do not only consider what happens in the resulting state of an action, but also what happens during its execution. For instance, van der Mayden in [27] defines a variation of dynamic deontic logic in which the semantics of deontic operators is given by means of traces instead of relations, the main aim of this system is to provide an alternative definition of permission, which, in Meyer's logic, exhibits some undesirable properties; for instance, the sentence: *if after shooting the president, one is allowed to remain in silence, then it is allowed to shoot the president and remain in silence* is valid, and contradicts our intuition about permission, this paradox is related to the operation of action composition, written formally as $\alpha;\beta$, van der Mayden's logic solves these issues of the original definition of permission in DDL, the resulting logic is interesting for dealing with composition of actions and action iteration, it is worth remarking that here we do not deal with action composition and related constructs. On the other hand, Dignum et al. [9] introduced two variants of permission to solve the *paradox of free choice*, which can be stated as: *if one is allowed to talk to the president, then one is allowed to talk the president or kill him*; a rich set of properties is provided for this system but completeness is not proven. Another approach worth mentioning is that presented in [14], the authors introduce an extension of dynamic deontic logic to deal with long term duties, of the style *Don't ever do that*, interestingly, the usual duality between permission and obligation that holds in dynamic deontic logic is not longer valid in this logic.

In a seminal paper [22], Segerberg introduced a different approach to the logical analysis of norms applied to practitions, the logic proposed by Segerberg (named DAL from now on) defines permission, prohibition and obligation without resorting to other logical constructions. This logic provides a set of basic actions that can be combined by using boolean operations; it is important to remark that DAL does not include the modalities of dynamic logic; this, in our opinion, simplifies the reasoning about deontic operators and the logic itself, thus a main benefit of Segerberg's approach is the simplicity of its formal system: its action algebra is based on the well-known theory of boolean algebras, which allows one to capture basic ways of combining actions (as pointed out above), and to reason about the properties of norms when applied to these constructions. Segerberg's logic inspired other systems, for instance, those proposed by Trypuz and Kulicki [26] (described in Section 2) and the system proposed by Castro and Maibaum [7]. It is important to remark that Segerberg's original system considers an enumerable number of basic actions, whereas the systems described in [26, 7] only consider a finite number of actions, this restriction has as a consequence that the associated boolean algebra of actions is finite, and thus atomic; this has some benefits, particularly, this makes it possible to refer to the basic constituent actions, thus, we have a set of basic actions that can be used to build any other action; this bears a direct analogy with computing systems in which a set of basic instructions, or actions, are provided and from them more complex programs can be constructed using programming constructs.

Axiomatic systems for DAL and its extensions were introduced in [26, 7], and these systems were shown to be sound and complete; however, we are not aware of any tableau system encompassing all these logics, a tableau system for the logic introduced in [7] was presented in [6], but this deductive system takes into account the Kripke semantics of the logic, and then it is more complex that the ones presented in this article. Furthermore, the computational complexity of DAL has not been investigated in the literature.

In this paper we present tableau systems for all the logics described in [26], that extend Sergerberg' original logic (and include the system introduced in [7]). We also show that the computational complexity of the satisfiability problem for these logics is NP-complete when a fixed number of actions are considered, and it is Σ_2^p-Hard when the set of actions in the language is taken as a parameter. In addition, we prove that the proof systems proposed below are sound and complete. Tableau systems [11] have been shown to be useful for providing automated methods of reasoning for several logics, some examples are propositional logic [25], first-order logic [11] and temporal logics [10]. Because of this, we believe that automated proof systems for DAL and its variations can be useful for applying deontic action logics in practice.

The article is organized as follows. In Section 2 we introduce the basic concepts needed to tackle the rest of the paper. In Section 3 we investigate the computational complexity of the satisfiability problem for DAL (and its variations). The tableau calculi for the deontic action logics considered throughout this text are described in Section 4 and 5; finally, we discuss some conclusions and further work.

2. Background

In this section we present the basic logics we use throughout this paper, the formalisms introduced below are strongly inspired by the deontic logic of actions presented by Segerberg in [22]. Given a finite set of *primitive* or *basic* actions $\Delta_0 = \{a_0, \dots, a_n\}$, the set of formulas of DAL is described by the following BNF:

$$\begin{aligned} \Phi &::= \mathbf{P}(\alpha) \mid \mathbf{F}(\alpha) \mid \Phi \vee \Phi \mid \neg\Phi \mid \top \\ \alpha &::= a_i \mid \alpha \sqcap \alpha \mid \overline{\alpha} \mid \alpha \sqcup \alpha \mid 0 \mid 1 \end{aligned}$$

We use letters $\alpha, \beta, \dots$ as variables over actions, and letters $\varphi, \psi, \dots$ as variables over formulas. The set of all action terms is denoted by Δ.

Intuitively, $\mathbf{P}(\alpha)$ says that *action α is allowed to be executed*, and $\mathbf{F}(\alpha)$ says that *executing α is forbidden*, the rest of the formulas have the standard semantics. We call formulas of type $\mathbf{P}(\alpha)$, $\mathbf{F}(\alpha)$ (and their negations) *basic deontic formulas*.

At this point, some words about our understanding of actions, and action execution, are useful. Given an action α, we assume that there may be many ways of executing it; consider, for instance, the action of *driving*, one may drive while smoking, or perhaps while talking by phone; thus, one may consider different scenarios in which an action can be performed. From our point of view, these different ways in which an action can be executed are due two things: firstly, because the combination of primitive actions by means of parallel execution, and, secondly, since the non-determinism inherent in the logic. Note that, in DAL, any action can be described as a non-deterministic combination of a parallel execution of primitive actions. It is important to remark that we have not introduced the obligation operator here, the semantics of obligation is not as straightforward as those of permission and prohibition, we will not deal with the obligation operator in this paper, but this operator can usually be reduced to the other deontic operators [22, 26, 7].

Let us introduce the semantics of DAL, there are different ways of defining the semantics for this logic, here we follows [26]. First we introduce the notion of *deontic structure* (or *deontic model*); intuitively, each action is interpreted as a set of basic events, these are abstract semantic entities that identify the occurrence of actions. A deontic structure is a tuple $\langle \mathcal{D}, \mathcal{I} \rangle$ where $\mathcal{D} = \langle \mathcal{E}, \mathit{Leg}, \mathit{Ill} \rangle$, and $\mathcal{E} = \{e_1, \ldots, e_n\}$ is a finite collection of events; *Leg* represents the collection of legal (or permitted) outcomes, and *Ill* denotes the collection of illegal (or prohibited) outcomes. Here, an outcome is a collection of events, and then $\mathit{Leg} \subseteq 2^{\mathcal{E}}$ and $\mathit{Ill} \subseteq 2^{\mathcal{E}}$. We require that $\mathit{Leg} \cap \mathit{Ill} = \{\emptyset\}$, i.e., there is no outcome that can be both allowed and forbidden. In addition, *Leg* and *Ill* must be ideals in the boolean algebra $\langle 2^{2^{\mathcal{E}}}, \emptyset, 2^{\mathcal{E}}, \cup, \cap, \setminus \rangle$, i.e., they must satisfy the following conditions:

L1 $s \in \mathit{Leg} \wedge s' \subseteq s \Rightarrow s' \in \mathit{Leg}$,

L2 $s \in \mathit{Leg} \wedge s' \in \mathit{Leg} \Rightarrow s \cup s' \in \mathit{Leg}$,

I1 $s \in \mathit{Ill} \wedge s' \subseteq s \Rightarrow s' \in \mathit{Ill}$,

I2 $s \in \mathit{Ill} \wedge s' \in \mathit{Ill} \Rightarrow s \cup s' \in \mathit{Ill}$,

We also consider an interpretation function $\mathcal{I} : \Delta_0 \to 2^{\mathcal{E}}$, which is extended to the set Δ of actions, as follows:

- $\mathcal{I}(0) = \emptyset$
- $\mathcal{I}(1) = \mathcal{E}$
- $\mathcal{I}(\alpha \sqcup \beta) = \mathcal{I}(\alpha) \cup \mathcal{I}(\beta)$
- $\mathcal{I}(\alpha \sqcap \beta) = \mathcal{I}(\alpha) \cap \mathcal{I}(\beta)$
- $\mathcal{I}(\overline{\alpha}) = \mathcal{E} \setminus \mathcal{I}(\alpha)$

Furthermore, we require that the interpretation of the atomic actions (in the sense given by boolean algebras) of the logic must be deterministic. More formally, the axioms of boolean algebra [24] induce a quotient boolean algebra of action terms (where each action term is an equivalence class of action terms), this boolean algebra is finite and thus atomic, the atoms are (equivalent to) monomials of the form $a_0^* \sqcap a_0^* \sqcap \dots \sqcap a_n^*$ where each a_i^* is either a_i or $\overline{a_i}$; intuitively, these actions represent the basic actions that can be performed by the system; further intuitions about this model of actions can be found in [26, 7]. The atomic actions over a vocabulary Δ_0 are denoted by letters $\delta_0, \dots, \delta_k$. An interesting element of this collection is action $\overline{a_0} \sqcap \dots \sqcap \overline{a_n}$, this term denotes the occurrence of an external action, that is, no action from the system is executed in this case; we denote this action by $\widehat{\delta}$. The set of atomic action terms preceding an action is defined as $At_{\sqsubseteq\alpha}(\Delta_0) = \{\delta_i \mid \mathbf{BA} \vdash \delta_i \sqsubseteq \alpha\}$, where $\mathbf{BA}$ is a complete set of axioms for boolean algebra, and $\vdash$ is the relation of provability. When Δ_0 is clear from context we just write $At(\alpha)$. Furthermore, we require:

$$\mathcal{I}(\delta_i) \leq 1, \tag{I}$$

that is, each atomic action when executed "produces" at most a unique event, this means that we assume that atomic actions are deterministic. It is worth noting that, in this setting, non-determinism arises as a consequence of the combination of several atomic actions.

Given a deontic structure M, the relation $\vDash$ of satisfiability between models and formulas is defined as follows:

- $M \vDash \mathbf{P}(\alpha) \Leftrightarrow \mathcal{I}(\alpha) \in Leg$,
- $M \vDash \mathbf{F}(\alpha) \Leftrightarrow \mathcal{I}(\alpha) \in Ill$,
- $M \vDash \top$,
- $M \vDash \alpha = \beta \Leftrightarrow \mathcal{I}(\alpha) = \mathcal{I}(\beta)$,
- $M \vDash \neg\varphi \Leftrightarrow M \nvDash \varphi$,
- $M \vDash \varphi \vee \psi \Leftrightarrow M \vDash \varphi$ or $M \vDash \psi$.

Given a formula φ, we say that $\vDash \varphi$ if $M \vDash \varphi$ for every model M.

We call this basic system DAL^0, and it is equivalent[1] to the *basic deontic logic of urn model action* introduced in [22], adding further restrictions we can obtain the following logical systems [26]. DAL^1 is DAL^0 extended with condition:

$$\forall a_i \in \Delta_0 : \mathcal{I}(a_i) \in Leg \vee \mathcal{I}(a_i) \in Ill, \tag{II}$$

that is, in this system each basic action is allowed or forbidden, this is sometimes called *the closure property* in jurisprudence: *what is not forbidden*

[1]When only finite vocabularies are considered.

is permitted. This logic is equivalent to the *basic deontic closed logic of urn model action* of Segerberg [22]. It is matter of discussion whether this axiom capture closeness, one may devise a scenario where smoking is allowed, but smoking inside a building is not, thus the formula $\mathbf{P}(smoking) \vee \mathbf{F}(smoking)$ does not hold in this situation.

DAL^2 is DAL^1 extended with condition:

$$(\mathcal{E} \setminus \bigcup_{a_i \in \Delta_0} \mathcal{I}(a_i)) \in Leg \vee (\mathcal{E} \setminus \bigcup_{a_i \in \Delta_0} \mathcal{I}(a_i)) \in Ill, \tag{III}$$

DAL^3 is DAL^1 extended with condition:

$$\mathcal{I}(a_1) \cup \cdots \cup \mathcal{I}(a_n) = \mathcal{E}, \tag{IV}$$

roughly speaking, the events that are considered in these logics are exactly those "produced" by the basic actions. In this case, we call these structures *closed* since they do not take into account events that occur outside of the system being described. Let us illustrate these ideas with a simple example, consider an automatic teller machine, then, in this case, you describe the system by providing the basic actions in these class of devices: withdraw money, deposit money, etc. But actions do not belonging to the teller are usually not taken into account, an example is an electrical overload, this event is not part of the system, and then it is not considered in the given description. Of course, the vocabulary can be enlarged to consider further actions.

DAL^4 is DAL^0 extended with the atom closure axiom:

$$\mathbf{P}(\delta_i) \vee \mathbf{F}(\delta_i) \tag{V}$$

for any atom δ_i of the boolean algebra of terms. Finally, we have the system DAL^5 that is DAL^4 extended with condition IV.

Some comments are useful about this system, the condition above says that any elemental action (that is, a parallel execution of basic actions) is either forbidden or allowed. For instance, if we consider the actions of driving and drinking, then might be the case that driving is neither allowed nor forbidden, however, driving while drinking should be either permitted or prohibited; this condition seems acceptable from the point of view of normativeness, in particular if one adheres to the principle of closeness.

3. Computational Complexity

Let us investigate the computational complexity of the satisfiability problem (abbreviated SAT from now on) for the logics described in Section 2. Note that tableau proof systems are basically procedures for model finding (to prove a formula we apply the procedure to its negation and then, if no model is found, the formula is considered a theorem), thus the complexity of SAT for DAL becomes relevant for evaluating the efficiency of the proof systems described later on. First, it is important to note that the number of actions in Δ_0 is relevant when checking the satisfiability of a formula φ, for instance formula: $\mathbf{P}(a \sqcup \overline{a}) \wedge \neg \mathbf{P}(a) \wedge \neg \mathbf{P}(\overline{a})$ is unsatisfiable for $\Delta_0 = \{a\}$, this mainly since for this vocabulary we can have two events, one for indicating the execution of a, and another for pointing out the occurrence of $\overline{a}$, if more actions are considered then the formula becomes satisfiable (as the reader can check). Below we provide two results: if the vocabulary is considered fixed, then SAT is NP-Complete for any version of DAL, but if we consider the number of actions as an extra parameter (i.e., the problem is, given a formula φ and a vocabulary Δ_0, we want to know whether φ is SAT for vocabulary Δ_0) SAT becomes Σ_2^p-Hard in Stockmeyer's hierarchy [3]. It is interesting to note that SAT is EXPTIME-complete for dynamic logics (and so for Dynamic Deontic Logics) [21]. The lower computational complexity of DAL is due the simplicity of its semantics, wherein no Kripke structures are present.

THEOREM 3.1. *Given a fixed vocabulary Δ_0, SAT is NP-Complete for* DAL^i *(for any i).*

Proof. *Proving that DAL^0 is NP-hard is direct. Given any CNF formula in propositional logic with variables $x_0, \dots, x_n$ we consider a vocabulary $\Delta_0 = \{x_0, \dots . x_n\}$ of primitive actions, and translate each propositional formula x_i to a deontic formula $\mathbf{P}(x_i)$, and $\neg x_i$ to $\neg \mathbf{P}(x_i)$, while the boolean operators are translated by applying the identity. We denote by $\tau(\varphi)$ the translated formula. Let us prove that φ is SAT iff $\tau(\varphi)$ is SAT. Suppose that v is a valuation, then we form the following action term: $\delta = x_0^* \sqcap x_1^* \sqcap \cdots \sqcap x_n^*$ where $x_i^* = x_i$ if $v(x_i) = \top$, and $x_i^* = \overline{x_i}$ otherwise. Now, consider the structure $M = \langle \mathcal{D}, \mathcal{I} \rangle$ with $\mathcal{D} = \langle \{\delta\}, \{\{\delta\}, \emptyset\}, \emptyset \rangle$, and $\mathcal{I}(a_i) = \{\delta\}$ if $\delta \sqsubseteq a_i$, and $\mathcal{I}(a_i) = \emptyset$, otherwise. it is direct to see that $M \vDash v(\varphi)$. For the other direction, assume $M \vDash \tau(\varphi)$ for some M, then we define a valuation v as follows: $v(x_i)$ iff $M \vDash \mathbf{P}(x_i)$, it is direct to prove that $v \vDash \varphi$. Furthermore, note that this proof works for any version of* DAL^i.

Let us now prove that DAL^0 is in NP. Note that, given an atom δ_i and an action term α we can check $\delta_i \sqsubseteq \alpha$ in linear time. An atom is simply a term $a_1^* \sqcap \cdots \sqcap a_n^*$ where a_i^* is either a_i or $\overline{a_i}$; the procedure for checking $\delta_i \sqsubseteq \alpha$ is recursive: the base cases are given by $\alpha = a_i$ in this case $\delta_i \sqsubseteq a_i$ is true when a_i does not appear negated in δ_i and false otherwise, thus a *simple inspection of δ_i is enough. If $\alpha = \overline{\beta}$ then $\delta_i \sqsubseteq \alpha$ if it is not the case that $\delta_i \sqsubseteq \beta$, which is checked recursively. If $\alpha = \alpha' \sqcap \alpha''$, then $a_i \sqsubseteq \alpha' \sqcap \alpha''$ when $\delta_i \sqsubseteq \alpha'$ and $\delta_i \sqsubseteq \alpha''$, the two can be checked applying recursion, for $\alpha = \alpha' \sqcup \alpha''$ the procedure is similar. It is direct to see that the time needed for checking $\delta_i \sqsubseteq \alpha$ with this procedure is linear with respect to the length of α.*

Now, we describe a non-deterministic procedure for checking satisfiability, given a formula φ we proceed as follows:

1. *For each sub formula $\neg\mathbf{P}(\alpha)$ or $\neg\mathbf{F}(\alpha)$ in φ we guess an element of $\{\delta_0, \dots, \delta_m, 0\} = At(\Delta_0) \cup \{0\}$, and we store $\langle\neg\mathbf{P}(\alpha), x\rangle$ or $\langle\neg\mathbf{F}(\alpha), x\rangle$ where $x \in At(\Delta_0) \cup \{0\}$, to keep track of the atoms guessed.*
2. *For the set of equations appearing in φ (denoted by $Eq(\varphi)$), we choose (nondeterministically) a collection $eqs = \{eq_0, \dots, eq_n\}$ of equations.*
3. *For each δ_i guessed, we calculate the set $equiv(\delta_i) = \{\delta_j \mid \exists \alpha = \beta \in els \wedge \delta_i \sqsubseteq \alpha \wedge \delta_j \sqsubseteq \beta\}$, this can be done in polynomial time w.r.t. the formula size, since the number of $\delta's$ does not depend on the formula, and the number of equation equations is bounded by $|\varphi|$, in addition checking $\delta_i \sqsubseteq \alpha$ is linear in the size of α. After that, we proceed as follows:*
   - *If $\varphi = \mathbf{P}(\alpha)$ we search for a tuple $\langle\neg\mathbf{P}(\beta), \delta_k\rangle$ such that $\delta_i \sqsubseteq \beta$ for some $\delta_i \in equiv(\delta_k)$, in such a case we return $false$, otherwise we return true.*
   - *If $\varphi = \mathbf{F}(\alpha)$ we search for a tuple $\langle\neg\mathbf{F}(\beta), \delta_k\rangle$ such that $\delta_i \sqsubseteq \alpha$ for some $\delta_i \in equiv(\delta_k)$, in such a case we return $false$, otherwise we return true.*
   - *If $\varphi = \neg\mathbf{P}(\alpha)$, we look for the tuple $\langle\neg\mathbf{P}(\alpha), \delta_i\rangle$ and check $\delta_i \sqsubseteq \alpha$, if this does not hold we return $false$, otherwise we return true,*
   - *If $\varphi = \neg\mathbf{F}(\alpha)$, we look for the tuple $\langle\neg\mathbf{F}(\alpha), \delta_i\rangle$ and check $\delta_i \sqsubseteq \alpha$, if this does not hold we return $false$, otherwise we return true,*
   - *If $\varphi = \alpha = \beta$ and this equation is in eqs, then we return true, otherwise we return $false$,*
   - *If $\varphi = \varphi_0 \vee \varphi_1$, we apply the algorithm to φ_0 and φ_1 and return true if someone of them returns true, otherwise we return $false$,*
   - *If $\varphi = \neg\varphi$, we apply the algorithm to φ and return true if the algorithm returns $false$ and viceversa.*

We only need to prove that this procedure is correct, that is: $SAT(\varphi)$ returns true iff there is a model M such that $M \vDash \varphi$. Suppose that $SAT(\varphi)$ returns *true, we build a model as follows. First, we define the collection of sets:*

$$per = \{\{\delta_i\} \mid \textit{ there is no pair } \langle \neg \mathbf{P}(\alpha), \delta_k \rangle \textit{ with } \delta_k \in equiv(\delta_i)\}$$

Roughly speaking, this set contains all the atoms that should be allowed, similarly we define:

$$forb = \{\{\delta_i\} \mid \textit{ there is no pair } \langle \neg \mathbf{F}(\alpha), \delta_k \rangle \textit{ with } \delta_k \in equiv(\delta_i)\}$$

The collection of atoms that must be forbidden. Finally, we define the model as follows: $D = \langle\langle \mathcal{E}, Leg, Ill\rangle, \mathcal{I}\rangle$,

- $\mathcal{E} = \{\delta_i \mid \delta_i \in \Delta_0\}$,
- $Leg = Id(per)$,
- $Ill = Id(forb)$,
- $I(a_i) = \{\delta_i \mid \delta_i \in At_{\sqsubseteq a_i}(\Delta_0/eqs(\varphi))\}$

Where $Id(S)$ *is the ideal generated by the collection* S, $\Delta_0/eqs(\varphi)$ *is the boolean algebra obtained from vocabulary* Δ_0 *modulo equations* $eqs(\varphi)$ *and* $At_{\sqsubseteq a_i}(\Delta_0/eqs(\varphi))$ *is the set of atoms in this algebra preceding* a_i. *We need to prove* $M \vDash \varphi$. *The proof is by induction, the base cases are basic deontic formulas*

By contradiction suppose that $M \nvDash \varphi$, *and* φ *is a basic deontic formula, if* $\varphi = \mathbf{P}(\alpha)$ *note that* $\mathbf{P}(\alpha)$ *is false in* M *since* $\alpha \notin Leg$, *and by definition this happens when we have some pair* $\langle \delta_i, \neg\mathbf{P}(\beta)\rangle$ *but then the algorithm cannot return true, and by contradiction we obtain* $D \vDash \varphi$. *If* $\varphi = \alpha = \beta$, *since* $\alpha = \beta \in eqs(\varphi)$ *we have* $\mathcal{I}(\alpha) = \mathcal{I}(\beta)$ *(they have the same atoms in* Δ_0/eqs*), for the other basic deontic formulas are similar and the inductive cases are direct by applying the inductive hypothesis, so* $M \vDash \varphi$.

For the other side, suppose that there is a model $M' \vDash \varphi$, *for this* M' *we have that a set of equations in* $Eq(\varphi)$ *are true, say eqs, and also also some subformulas of the form* $\varphi_i = \neg\mathbf{P}(\alpha_i)$ *(or* $\varphi_i = \neg\mathbf{F}(\alpha_i)$*) such that* $M' \vDash \varphi_i$, *if this collection of sub formulas is empty, then note that the algorithm should have guessed eqs and returned true. Then the collection of* φ_i *must be non-empty, now for each subformula* $\neg\mathbf{P}(\alpha_i)$ *we have an atom* $\delta_j \sqsubseteq \alpha_i$ *such that* $M' \vDash \neg\mathbf{P}(\delta_i)$ *these* δ_i*'s can be guessed by the algorithm above, and if the algorithm does not return true for these* δ*'s is since any sub formula of* φ *has a sub formula* $\mathbf{P}(\alpha)$ *such that* $\delta_i \sqsubseteq \alpha$ *and* δ_i *is in the set guessed, and so* $M' \vDash \mathbf{P}(\delta_i)$, *this is a contradiction, then the algorithm must return true.*

For the other logics the proof is similar. For DAL^1 *we can add the formula* $\mathbf{P}(a_i) \vee \mathbf{F}(a_i)$ *for each* a_i *in* φ. *The length of this formula does not depend* *on φ and so it does not affect the efficiency of the algorithm discussed above. For* DAL^2 *we add the formula* $\mathbf{P}(\overline{a_0} \sqcap \dots \sqcap \overline{a_n}) \vee \mathbf{F}(\overline{a_0} \sqcap \dots \sqcap \overline{a_n})$ *to φ. For* DAL^3 *we add the equation* $a_0 \sqcup \dots \sqcup a_n = 1$ *to eqs. For* DAL^4 *we add formulas* $\mathbf{P}(\delta_i) \vee \mathbf{F}(\delta_i)$ *to φ, as before note that the length of this formula do not depend on φ, and similarly for* DAL^5*. The result follows.*

If we consider vocabularies as an extra parameter of the decision method the complexity of the SAT problem gets "harder", this is proven in the following theorem.

THEOREM 3.2. *Given a formula φ and a vocabulary Δ_0, the problem of deciding whether φ is satisfiable for some model of Δ_0 is Σ_2^p-Hard for* DAL^i *(for any i).*

Proof. *We reduce a complete problem for Σ_2^p to DAL satisfiability, in particular we use Quantified Boolean Formulas (QBF), the reader is referred to [3] for the details about this logic. Consider quantified boolean formulas of the style:*

$$\exists x_0, \dots, x_n : \forall y_0, \dots, y_m : \varphi, \tag{VI}$$

where the x_i's and y_j's are propositional variables, and φ is a boolean formula in conjunctive normal form in which only quantified variables occur. Deciding the truth of these kinds of formulas is a problem that is Σ_2^p-Complete [3]. Given a quantified boolean formula ϕ, we define a DAL formula $\tau(\phi)$ (note that the proof below works for any version of DAL). Assume that ϕ has the form of formula VI; first, we define the vocabulary:

$$\Delta_0 = \{X_0, \dots, X_n, x_0, \dots, x_n, Y_0, \dots, Y_m, \neg Y_0, \dots, \neg Y_m\},$$

where $x_0, \dots, x_n$ are the variables that appear existentially quantified in ϕ, and the rest are fresh symbols. First we define a mapping from boolean literal to action terms as follows:

- $\tau(x_i) = X_i$,
- $\tau(\neg x_i) = \overline{X_i}$,
- $\tau(y_i) = Y_i$,
- $\tau(\neg y_i) = \neg Y_i$.

Let us define some auxiliary formulas.

$$\Phi_\forall = \bigwedge_{Y_i} \mathbf{F}(Y_i \sqcap \neg Y_i) \wedge \mathbf{F}(\overline{Y_i} \sqcap \overline{\neg Y_i}) \wedge Y_i \neq \emptyset \wedge \neg Y_i \neq \emptyset \wedge \mathbf{P}(Y_i \sqcup \neg Y_i) \tag{VII}$$

Roughly speaking, this formula states that some way of executing both Y_i and $\neg Y_i$ must be allowed. On the other hand, we have another formula for existentially quantified variables.

$$\Phi_{\exists} = \bigwedge_{x_i}((X_i = x_i \vee X_i = \overline{x_i}) \wedge X_i \neq \emptyset \wedge \mathbf{P}(x_i) \Rightarrow \mathbf{F}(\overline{x_i}) \wedge \mathbf{P}(\overline{x_i}) \Rightarrow \mathbf{F}(x_i)) \tag{VIII}$$

This formula says that any occurrence of X_i can be replaced by either x_i or $\overline{x_i}$, and only one of them is allowed.

Now, we can define the translation of the boolean formula (recall that ϕ is in CNF form).

- $\tau(z_0 \vee \cdots \vee z_k) = \mathbf{P}(\tau(z_0) \sqcup \cdots \sqcup \tau(z_k)) \wedge \mathbf{F}(\overline{\tau(z_0)} \sqcap \cdots \sqcap \overline{\tau(z_k)})$,
- $\tau(c_0 \wedge \cdots \wedge c_t) = \tau(c_0) \wedge \cdots \wedge \tau(c_t)$

and for the complete formula te translation is as follows:

$$\tau(\phi) = \Phi_{\exists} \wedge \Phi_{\forall} \wedge \tau(\varphi(x_0, \ldots, x_n, y_0, \ldots, y_m)))$$

Intuitively, each atom in the term algebra denotes an assignment to variables, for instance, the atom $x \sqcap \overline{y}$ denotes the assignment in which x is true and y is false. The main idea is that atomic action terms representing satisfying assignments will belong to the set Leg in a DAL *model.*

First observe that, for any QBF φ, the size of $\tau(\varphi)$ is linearly bounded by the size of φ, that is, the translation can be done in polynomial time. We now need to prove that this translation is correct, that is, φ is true iff $\tau(\varphi)$ is SAT.

Let us prove the implication from left to right first, suppose that ϕ is true, recall that the boolean part is a CNF formula, that is, it is a conjunction of clauses $\phi_0 \wedge \cdots \wedge \phi_k$. Consider valuation $v : \{x_0, \ldots, x_n\} \to Bool$ that makes formula ϕ true, we build a model of $\tau(\phi)$ as follows:

$M = \langle\langle \mathcal{E}, Leg, Ill\rangle, \mathcal{I}\rangle$, where $\mathcal{E} = At(\Delta_0)$; for defining the sets Leg and Ill we consider the following auxiliar function for every $a \in \Delta_0$ and $\delta \in \mathcal{E}$:

$$val(a)(\delta) = \begin{cases} true & \text{if } a \text{ does not appear negated in } \delta \\ false & \text{otherwise} \end{cases}$$

that is, this function says if a primitive action appears negated or not in a given atomic action term. Using val we can define the following collection of sets:

$$S = \{\{\delta\} \mid \delta \in \mathcal{E} \wedge \forall x_i : \varphi[x_i := val(\delta)(x_i)] = true \wedge \forall y_i : val(Y_i) \neq val(\neg Y_i)\} \tag{IX}$$

and we also consider:

$$\overline{S} = \{\{\delta\} \mid \delta \in \mathcal{E} \wedge \{\delta\} \notin S\}$$

Then we define:

$$Leg = Id(S)$$

the ideal generated by S, furthermore:

$$Ill = Id(\overline{S}).$$

that is the ideal generated by the complement of S. Furthermore, for each $a \in \Delta_0$ we define $\mathcal{I}(a) = At(a) \cap \bigcup S$.

Rest to prove that $M \vDash \tau(\phi)$, let us prove that $M \vDash \Phi_\forall \wedge \Phi_\exists \wedge \tau(\phi_i)$, for every clause ϕ_i, we know that $\phi_i = \ell_0 \vee \cdots \vee \ell_k$, that is: $\tau(\phi_i) = \mathbf{P}(\tau(\ell_0) \sqcup \cdots \sqcup \tau(\ell_k)) \wedge \mathbf{F}(\overline{\tau(\ell_0)} \sqcap \cdots \sqcap \overline{\tau(\ell_k)})$ this could be false since there is a $\delta \in \mathcal{I}(\tau(\ell_0) \sqcup \cdots \sqcup \tau(\ell_k))$ such that $\{\delta\} \notin Leg$, by definition this only may happen when $M \vDash \tau(\ell_0) \sqcup \cdots \sqcup \tau(\ell_k) = \emptyset$, and this by Definition IX implies that there is no boolean values $v_0, \dots, v_n$ such that $\phi_i[x_0 := v_0, \dots, x_n := v_n]$ is true, but this contradicts our initial assumption that ϕ is true, and then we have $M \vDash \phi$.

For the other direction, suppose $M \vDash \tau(\phi)$, we define an assignment v from existentially quantified variables to boolean values, for defining v we take any atomic term δ such that $M \vDash \mathbf{P}(\delta) \wedge \delta \neq \emptyset$, there must be at least one of such terms, otherwise $Leg = \{\emptyset\}$ and ϕ cannot be true in the model since $\Phi_\exists$ and $\Phi_\forall$ would be false. Then, we define $v(x_i) = true$ iff x_i appears positively in δ. Now we assert that $\phi[x_0 := v(x_0), \dots, x_n := v(x_n)] = true$, let us prove this, if $\phi[x_0 := v(x_0), \dots, x_n := v(x_n)] = false$ that is, there is a clause in which all literals are false on this valuation, suppose $\phi_i = \ell_0 \vee \cdots \vee \ell_k$ is such a clause we have that all the literals that are x's (or their negations) are false under v, thus if $x = \ell_i$ for some i, then x does not appear in δ, and if $\neg x = \ell_i$ for some i then x appears positively in δ; if we change, in this action term, any y_i occurring in δ by $\overline{y_i}$ (its negation), since $\Phi_\forall$ we should have that $M \vDash \mathbf{P}(\delta')$ where δ' is the resulting term, but we should have that $M \vDash \mathbf{F}(\delta')$ (because the definition of $\tau(\phi_i)$), and this contradicts $\Phi_\forall$, and therefore $M \nvDash \phi$ but this is a contradiction, the result follows.

4. The Basic Tableau for DAL^0

Now we present a tableau for DAL^0, we extend this system in Section 5 to cope with the other logics. The tableau rules for the boolean operators are standard, and are shown in Figure 5. We follow Smullyan's terminology [25] and we classify DAL^0 formulas into A, B, C, and D classes of formulas; the first ones are formulas that can be thought of as being conjunctions (Figure 1); formulas of type B are those that can be identified with disjunctions (Figure 2); formulas C are those corresponding to deontic operators applied to conjunctions of actions (Figure 3), whilst formulas D are negations of basic deontic formulas (Figure 4). The tableau method takes as parameters a formula to be proven (say φ) and the vocabulary over which the formula is defined (say Δ_0). The notions of *tableau* and *branch* are standard and

A	A_1	A_2
$\varphi \wedge \psi$	φ	ψ
$\neg(\varphi \vee \psi)$	$\neg\varphi$	$\neg\psi$
$\mathbf{P}(\alpha \sqcup \beta)$	$\mathbf{P}(\alpha)$	$\mathbf{P}(\beta)$
$\mathbf{F}(\alpha \sqcup \beta)$	$\mathbf{F}(\alpha)$	$\mathbf{F}(\beta)$
$\neg\mathbf{P}(\alpha \sqcap \beta)$	$\neg\mathbf{P}(\alpha)$	$\neg\mathbf{P}(\beta)$
$\neg\mathbf{F}(\alpha \sqcap \beta)$	$\neg\mathbf{F}(\alpha)$	$\neg\mathbf{F}(\beta)$
$\mathbf{P}(\overline{\alpha \sqcap \beta})$	$\mathbf{P}(\overline{\alpha})$	$\mathbf{P}(\overline{\beta})$
$\mathbf{F}(\overline{\alpha \sqcap \beta})$	$\mathbf{F}(\overline{\alpha})$	$\mathbf{F}(\overline{\beta})$

Figure 1. Formulas of Type A

B	B_1	B_2
$\varphi \vee \psi$	φ	ψ
$\neg(\varphi \wedge \psi)$	$\neg\varphi$	$\neg\psi$
$\neg\mathbf{P}(\alpha \sqcup \beta)$	$\neg\mathbf{P}(\alpha)$	$\neg\mathbf{P}(\beta)$
$\neg\mathbf{F}(\alpha \sqcup \beta)$	$\neg\mathbf{F}(\alpha)$	$\neg\mathbf{F}(\beta)$
$\neg\mathbf{P}(\overline{\alpha \sqcap \beta})$	$\neg\mathbf{P}(\overline{\alpha})$	$\neg\mathbf{P}(\overline{\beta})$
$\neg\mathbf{F}(\overline{\alpha \sqcap \beta})$	$\neg\mathbf{F}(\overline{\alpha})$	$\neg\mathbf{F}(\overline{\beta})$

Figure 2. Formulas of Type B

can be found in the literature [25, 11], for the sake of clarity, we recast them here.

Definition 4.1. A tableau is an (n-ary) rooted tree, where each node is labelled with a set of formulas and a branch is a path from the root to some leaf.

From now on, we identify any branch $\mathcal{B}$ with its set of formulas. An interesting observation is that formulas of the type $\neg\mathbf{P}(\alpha)$ (or $\neg\mathbf{F}(\alpha)$) have

C	$C(\delta_i)$
$\mathbf{F}(\alpha \sqcap \beta)$	$\mathbf{F}(\delta_i)$ for some $\delta_i \in Atom(\alpha) \cap Atom(\beta)$
$\mathbf{P}(\alpha \sqcap \beta)$	$\mathbf{P}(\delta_i)$ for some $\delta_i \in Atom(\alpha) \cap Atom(\beta)$
$\mathbf{P}(\overline{\alpha \sqcup \beta})$	for some $\delta_i \notin Atom(\alpha) \cap Atom(\beta)$
$\mathbf{F}(\overline{\alpha \sqcup \beta})$	for some $\delta_i \notin Atom(\alpha) \cap Atom(\beta)$
$\mathbf{F}(a_i)$	$\mathbf{F}(\delta_i)$ for some $\delta_i \in Atom(a_i)$
$\mathbf{P}(a_i)$	$\mathbf{P}(\delta_i)$ for some $\delta_i \in Atom(a_i)$
$\mathbf{F}(\overline{a_i})$	$\mathbf{F}(\delta_i)$ for some $\delta_i \notin Atom(a_i)$
$\mathbf{P}(\overline{a_i})$	$\mathbf{P}(\delta_i)$ for some $\delta_i \notin Atom(a_i)$

Figure 3. Formulas of Type C

$D(a_i)$	$D(\delta_i)$
$\neg\mathbf{F}(a_i)$	$\neg\mathbf{F}(\delta_i)$ for some $\delta_i \in Atom(a_i)$
$\neg\mathbf{P}(a_i)$	$\neg\mathbf{P}(\delta_i)$ for some $\delta_i \in Atom(a_i)$
$\neg\mathbf{F}(\overline{a_i})$	$\neg\mathbf{F}(\delta_i)$ for some $\delta_i \in Atom(a_i)$
$\neg\mathbf{P}(\overline{a_i})$	$\neg\mathbf{P}(\delta_i)$ for some $\delta_i \in Atom(a_i)$

Figure 4. Formulas of Type D

an "existential" character, in the sense that they state that there is some way of executing α which is not permitted (or not forbidden), these formulas imply that $\alpha \neq \emptyset$, thus there must be some way of executing α, this fact inspires the following definition:

DEFINITION 4.2. Given any branch $\mathcal{B}$, we define the following set:

$$\mathcal{B}^* = \{\alpha \neq 0 \mid \neg\mathbf{P}(\alpha) \in \mathcal{B} \text{ or } \neg\mathbf{F}(\alpha) \in \mathcal{B}\} \cup \mathcal{B}$$

Thus, $\mathcal{B}^*$ is the branch extended with a collection of inequations implied by the deontic formulas already in the branch. Furthermore, any branch introduces an equational theory:

DEFINITION 4.3. Given a branch $\mathcal{B}$, we define:

$$Eq(\mathcal{B}) = \{\alpha = \beta \mid \alpha = \beta \in \mathcal{B}\} \cup \{\alpha = 0 \mid \mathbf{P}(\alpha) \in \mathcal{B} \wedge \mathbf{F}(\alpha) \in \mathcal{B}\}$$

Consider the tableau rules shown in Figure 5. The tableau calculus for DAL^0 consists of rules $\mathbf{A}, \mathbf{B}, \mathbf{C}$, $\mathbf{D}$, **Ref** and **Eq**. Rules of type $\mathbf{A}$ and $\mathbf{B}$ are standard, more interesting for us are rules $\mathbf{C}$ and $\mathbf{D}$. Roughly speaking, rule $\mathbf{D}$ says that, if we have formula $\neg\mathbf{P}(a_i)$ in a branch, for some $a_i \in \Delta_0$, then there is some atom δ_i with $\delta_i \sqsubseteq a_i$, such that we have $\neg\mathbf{P}(\delta_i)$ in the branch,

$$\mathbf{A} : \frac{A}{\begin{array}{c}A_1\\A_2\end{array}} \qquad \mathbf{B} : \frac{B}{B_1 \quad B_2}$$

$$\mathbf{C} : \frac{C(\alpha \sqcap \beta)}{C(\delta_i)}[\textbf{Proviso 1}] \qquad \mathbf{D} : \frac{D(\alpha)}{D(\delta_0) \mid \cdots \mid D(\delta_n)}$$

$$\mathbf{Ref} : \frac{}{\alpha = \alpha} \qquad \mathbf{Eq} : \frac{\phi}{\phi[\alpha := \beta]}[\textbf{Proviso 2}]$$

Figure 5. Tableau Rules for DAL^0

i.e., this atom provides a witness for the formula. Note that for rule **C** we have the following proviso:

- **Proviso 1**: For some δ_i appearing in the branch with $\delta_i \in At(\alpha)$ and $\delta_i \in At(\beta)$

For instance, given $\mathbf{P}(\alpha \sqcap \beta)$ we can add to the actual branch the formula $\mathbf{P}(\delta_i)$, where δ_i is an atom of both α and β that is already in the branch, thus, it was added in the branch by the application of another rule. On the other hand, Rules **Ref** and **Eq** are standard rules for equality [11]; **Ref** is the rule for reflexivity, in any branch we may add the equation $\alpha = \alpha$ to the branch (we restrict the application of this rule to actions α appearing in the branch). Rule **Eq** is the replacement of equals by equals, we have the following proviso for this rule:

- **Proviso 2**: ϕ is a basic deontic formula or an equation, and equation $\alpha = \beta$ belongs to the actual branch.

That is, if we have a formula ϕ where an action α occurs, and equation $\alpha = \beta$ belongs to the branch, then we can add $\phi[\alpha := \beta]$ (the formula obtained by replacing some occurrences of α by β) to the branch. As described in [11] the other properties of equality can be proven from these rules. Note that, when constructing the tableau, standard rules are applied first, this implies that we have the set of equations occurring in the input formula at hand before we start to apply deontic rules; also note that **Eq** is directional, that is, it allows us to replace the left term in the equation by the right term, but not viceversa.

As usual we define the notions of *closed* and *open* branches, intuitively, the former captures those branches that are inconsistent, and the latter identifies the branches that are satisfiable. To this end, we define some conditions about sets for formulas. Given a set of DAL^0 formulas S over a vocabulary $\Delta_0 = \{a_0, \dots, a_n\}$, we consider the following conditions:

C0 $Eq(S) \vdash 1 = 0$,

C1 $\varphi \in S$ and $\neg\varphi \in S$,

C2 $Eq(S) \vdash \alpha = \beta$ and $\alpha \neq \beta \in S$,

C3 $Eq(\mathcal{S}) \cup \{1 = a_0 \sqcup \cdots \sqcup a_n\} \vdash \alpha = \beta$ and $\alpha \neq \beta \in \mathcal{S}$.

Now, we can define the notion of closed branch:

Definition 4.4. Given a branch $\mathcal{B}$ of a tableau, we say that $\mathcal{B}$ is closed for DAL^0 if either **C0**, **C1** or **C2** holds for $\mathcal{B}^*$,

Note that we do not have used condition **C3** in this definition, this condition is needed for the other versions of DAL.

Let us prove the soundness and completeness of the tableau calculus for DAL^0; towards this end we introduce the notions of *complete* and *SAT* branch.

Definition 4.5. We say that a branch $\mathcal{B}$ is complete for DAL^0 if:

- A formula A belongs to $\mathcal{B}$, then A_1 and A_2 belong to $\mathcal{B}$,
- A formula B belongs to $\mathcal{B}$, then either B_1 or $B2$ belong to $\mathcal{B}$,
- A formula D belongs to $\mathcal{B}$, then some $D(\delta_i)$ belongs to $\mathcal{B}$, and δ_i appears in other formula in the branch.
- A formula C belongs to $\mathcal{B}$, and for some $C(\delta_i)$, δ_i appears in other formula in the branch, then $C(\delta_i)$ belongs to $\mathcal{B}$.
- A basic deontic formula or an equation ϕ and an equation $\alpha = \beta$ belong to $\mathcal{B}$, such that α occurs in ϕ, then $\phi[\alpha := \beta]$ belongs to $\mathcal{B}$.

Intuitively, a branch is complete if all the possible rules have been applied. The following definition is needed for proving the soundness of the tableau calculus.

Definition 4.6. A branch $\mathcal{B}$ is SAT if there is a deontic structure M such that $M \vDash \mathcal{B}$.

In the definition above, recall that we identify branches with its collection of formulas, thus, a deontic structure satisfies a branch iff it satisfies all the formulas in the branch.

THEOREM 4.7. *Given a SAT branch $\mathcal{B}$, then any branch $\mathcal{B}'$ obtained by applying the rules for $\mathcal{DAL}^0$ is SAT.*
Proof. *Let $\mathcal{B}$ a branch, and $M = \langle \mathcal{D}, \mathcal{I} \rangle$ and $\mathcal{D} = \langle \mathcal{E}, Leg, Ill \rangle$.*

- *Rule* **A**, *the case of boolean formulas is straightforward. For $\mathbf{P}(\alpha \sqcup \beta)$, suppose $M \vDash \mathbf{P}(\alpha \sqcup \beta)$, that is, $\mathcal{I}(\alpha \sqcup \beta) \in Leg$, thus, $\mathcal{I}(\alpha) \in Leg$ and $\mathcal{I}(\beta) \in Leg$ which implies $M \vDash \mathbf{P}(\alpha)$ and $M \vDash \mathbf{P}(\beta)$. For $\mathbf{F}(\alpha \sqcup \beta)$ is the same. The proof for negated formulas is similar, for instance, if $M \vDash \neg\mathbf{P}(\alpha \sqcap \beta)$ then, by definition, $\mathcal{I}(\alpha \sqcap \beta) \notin Leg$, thus by condition* ***L1****, $\mathcal{I}(\alpha) \notin Leg$ and $\mathcal{I}(\beta) \notin Leg$, and then $M \vDash \neg\mathbf{P}(\alpha)$ and $M \vDash \neg\mathbf{P}(\beta)$.*
- *Rule* **B**, *again the proof for boolean formulas is direct. For $\neg\mathbf{P}(\alpha \sqcup \beta)$, if $M \vDash \neg\mathbf{P}(\alpha \sqcup \beta)$, then $I(\alpha \sqcup \beta) \notin Leg$, and therefore (by condition* **L2***) $I(\alpha) \notin Leg$ or $I(\beta) \notin Leg$, thus $M \vDash B_1$ or $M \vDash B_2$. The proofs for the other formulas are similar.*
- *Rule* **D**, *suppose $D \in \mathcal{B}$, if $M \vDash \neg\mathbf{P}(a_i)$, then $I(a_i) \notin Leg$, thus we must have some $\delta_i \in At(a_i)$ such that $I(\delta_i) = \{e\}$ and $\{e\} \notin Leg$ this implies that $M \vDash \neg\mathbf{P}(\delta_i)$.*
- *Rule* **C**, *the proof is similar to the case of rule D.*
- *The proof for rules* **Ref** *and* **Eq** *are standard and the reader is referred to [11].*

The soundness of the tableau calculus follows:

COROLLARY 4.8. *If φ is tableaux provable in* DAL^0 *(that is, there exists a closed tableau for φ), then $\vDash \varphi$.*

Towards the proof of completeness we introduce the notion of Hintikka sets.

DEFINITION 4.9. Let S be a set of formulas we say that S is Hintikka for DAL^0 iff:

- S is not closed for DAL^0,
- S is complete.

The interesting fact about Hintikka sets is that they are SAT.

THEOREM 4.10. *Any Hintikka set for* DAL^0 *is SAT.*
Proof. *Given a Hintikka set S we define a deontic structure. Let $Eq(S)$ all the equations in S. We denote by $\Delta_0/Eq(S)$ be the boolean action algebra of terms modulo $Eq(S)$, and $At(\Delta_0)$ the set of atoms in this algebra (denoted*

by $\delta_0, \delta_1, \ldots, \delta_n$). Recall that, for any action a_i we denote by $At(a_i)$ its set of atoms. Now, for each $a_i \in \Delta_0$ we define the following sets:

$$Per = \{\{\delta_i\} \mid \nexists \neg \mathbf{P}(\delta_i) \in \mathcal{B}\}$$

and similar for forbidden actions:

$$Forb = \{\{\delta_i\} \mid \nexists \neg \mathbf{F}(\delta_i) \in \mathcal{B}\}$$

We define the model $M^* = \langle\langle At(\Delta_0), Leg^*, Ill^*\rangle, \mathcal{I}\rangle$ where:

- *$Leg^* = Id(Per)$,*
- *$Ill^* = Id(Forb)$*

Thus, Leg^* is the ideal in the boolean algebra $\langle 2^{2^{\mathcal{E}}}, \emptyset, 2^{\mathcal{E}}, \cup, \cap, \backslash \rangle$ generated by the basis Per; and Ill^* is the ideal generated by the basis Forb. The interpretation function is defined as follows:

$$\mathcal{I}(a_i) = \{\delta_i \mid \delta_i \in At(a_i)\}$$

Now, we need to prove that M^* is a model for $\mathcal{B}$, that is, $M^* \vDash \mathcal{B}$. The proof is by induction. The base cases are formulas of the style $\mathbf{P}(\alpha)$ and $\mathbf{F}(\alpha)$. In this case we apply again induction on α, if $\alpha = a_i$ and $a_i \in \mathcal{B}$, suppose $M^* \nvDash \mathbf{P}(a_i)$, thus we have $\mathcal{I}(a_i) \notin Leg^*$, since $a_i = \delta_1 \sqcup \cdots \sqcup \delta_k$ where $\delta_i \in At(a_i)$, we must have $\mathcal{I}(\delta_i) \notin Leg^*$ by definition of Leg^* this only happens when $\neg\mathbf{P}(\delta_i) \in \mathcal{B}$, but then by application of Rule **C** *we must have $\mathbf{P}(\delta_i) \in \mathcal{B}$ and the branch is closed, which is a contradiction and the result follows. The proof for $\mathbf{F}(a_i)$ is similar. For the rest of the action terms the proofs are similar applying the corresponding rules and the inductive hypothesis.*

This theorem implies the completeness of the calculus.

THEOREM 4.11. *The tableau calculus for DAL^0 is complete, that is: $\vDash \varphi$ implies $\vdash \varphi$.*
Proof. *Suppose that $\nvdash \varphi$, that means that we have an open tableau for $\neg\varphi$, by theorem 4.10 this implies that we have a model $M^* \vDash \neg\varphi$ and so $\nvDash \varphi$, the result follows.*

5. Extensions of the Basic Tableau System

We extend the tableau shown above to cope with any DAL^i. Let us consider the additional rules shown in Figure 6. The tableau calculus for DAL^1 con-

$$\mathbf{R1a}\frac{\neg\mathbf{P}(\delta_i)}{\mathbf{F}(a_i)} \qquad \mathbf{R1b}\frac{\neg\mathbf{F}(\delta_i)}{\mathbf{P}(a_i)}$$

$$\mathbf{R2a}:\frac{\neg\mathbf{F}(\widehat{\delta_i})}{\mathbf{P}(\widehat{\delta_i})} \qquad \mathbf{R2a}:\frac{\neg\mathbf{P}(\widehat{\delta_i})}{\mathbf{F}(\widehat{\delta_i})}$$

$$\mathbf{R4a}:\frac{\neg\mathbf{F}(\delta_i)}{\mathbf{P}(\delta_i)} \qquad \mathbf{R4b}:\frac{\neg\mathbf{P}(\delta_i)}{\mathbf{F}(\delta_i)}$$

Figure 6. Tableau Rules for DAL^1, DAL^2, DAL^3, DAL^4, DAL^5

sists of rules of DAL^0 plus rules $\mathbf{R1a}, \mathbf{R1b}$. For DAL^2 we have the rules of DAL^0 plus rules $\mathbf{R2a}, \mathbf{R2b}$. For DAL^3 we have the same rules as for DAL^0. For DAL^4 we have the rules of DAL^0 and rules $\mathbf{R4a}, \mathbf{R4b}$. Note that DAL^3 has the same rules as DAL^0, the difference between these two systems is given by the notion of closed branch.

Definition 5.1. $\mathcal{B}$ is closed for $\mathsf{DAL}^3, \mathsf{DAL}^4$ and DAL^5 if either **C0**, **C1** or **C3** holds,

For the other versions of DAL the definition of closed branch is as in Definition 4.4. The notions of *complete branch* and *Hintikka branch* for these logics are direct.

Definition 5.2. Given a branch $\mathcal{B}$ we say that it is complete for DAL^1 or for DAL^3 if it is complete for DAL^0 and:

- If $\neg\mathbf{P}(\delta_i) \in \mathcal{B}$, then $\mathbf{F}(a_i) \in \mathcal{B}$ where $\delta_i \in At(a_i)$,
- If $\neg\mathbf{F}(\delta_i) \in \mathcal{B}$, then $\mathbf{P}(a_i) \in \mathcal{B}$ where $\delta_i \in At(a_i)$,

Definition 5.3. Given a branch $\mathcal{B}$ we say that it is complete for DAL^2 if it is complete for DAL^1 and:

- If $\neg\mathbf{P}(\widehat{\delta_i}) \in \mathcal{B}$, then $\mathbf{F}(\widehat{\delta_i}) \in \mathcal{B}$ where $\delta_i \in At(a_i)$,
- If $\neg\mathbf{F}(\widehat{\delta_i}) \in \mathcal{B}$, then $\mathbf{P}(\widehat{\delta_i}) \in \mathcal{B}$ where $\delta_i \in At(a_i)$,

Definition 5.4. Given a branch $\mathcal{B}$ we say that it is complete for DAL^4 and DAL^5 if it is complete for DAL^0 and:

- If $\neg\mathbf{P}(\delta_i) \in \mathcal{B}$, then $\mathbf{F}(\delta_i) \in \mathcal{B}$ where $\delta_i \in At(a_i)$,

- If $\neg\mathbf{F}(\delta_i) \in \mathcal{B}$, then $\mathbf{P}(\delta_i) \in \mathcal{B}$ where $\delta_i \in At(a_i)$,

The notion of Hintikka branch is the same as in Definition 4.9. The proofs of soundness for these tableau calculi can be obtained by extending both the definition of complete branch and the proof of Theorem 4.7 in a direct way.

THEOREM 5.5. *The tableau calculus for DAL^i (with $i \in \{1,2,3,4,5\}$) is sound.*

More interesting are the proof of completeness for these extensions.

THEOREM 5.6. *If $\mathcal{B}$ is a Hintikka branch for DAL^i, then there is a model M^* such that $M^* \vDash \varphi$.*
Proof. *First let us prove the property for DAL^1. We define the model M^* exactly as in Theorem 4.10; we just need to prove that M^* holds Condition II. Suppose that $M^* \vDash \neg\mathbf{P}(a_i)$ and $M^* \vDash \neg\mathbf{F}(a_i)$, by the definition of M^* this implies that $\neg\mathbf{P}(\delta_i) \in \mathcal{B}$ and $\neg\mathbf{F}(\delta_j) \in \mathcal{B}$ for some $\delta_i, \delta_j \in \mathcal{B}$, by rules* **R1** *and the definition of complete branch we have that $\mathbf{F}(a_i) \in \mathcal{B}$, and by rule **C**, we must have $\mathbf{F}(\delta_j) \in \mathcal{B}$, which implies that the branch is closed giving a contradiction. For DAL^2 the proof is similar, but using rules **R2a** and **R2b**. For DAL^3 we add the equation $a_0 \sqcup \cdots \sqcup a_n = 1$ to $Eq(S)$, and the result follows. The proof is more complex for DAL^4, in this case if $M^* \vDash \neg\mathbf{P}(\delta_i)$ and $M^* \vDash \neg\mathbf{F}(\delta_i)$, thus by definition of M^* we have that $\neg\mathbf{P}(\delta_i) \in \mathcal{B}$ and $\neg\mathbf{F}(\delta_i) \in \mathcal{B}$, but then by rules* **R4a** *and* **R4b** *we obtain $\mathbf{P}(\delta_i) \in \mathcal{B}$ and $\mathbf{F}(\delta_i) \in \mathcal{B}$ which implies that $\mathcal{B}$ is not Hinttika, giving us a contradiction.The proof for DAL^5 is similar.*

A standard argument can be used to prove that the tableau methods proposed above can be computed in PSPACE.

THEOREM 5.7. *For any formula in DAL, the tableau can be developed in PSPACE.*
Proof. *The proof is based on observing that each branch can be developed in polynomial space, and branches can be inspected one per time in a backtracking fashion. In any branch we have a collection of sub formulas of the given initial formula, first note that the number of sub formulas is polynomially bounded by the length of the formula, also we may add (by Rule **C**) new formulas to the branch, but we only can add one of these formulas for each application of rule **D** in the branch, and we have at most one application of this rule for each sub formula (which is polynomially bounded), then in each branch we can only add a polynomial amount of formulas of type $C(\delta_i)$,*

that is, the size of any branch is polynomially bounded by the size of the original formula, we inspect one branch per time, and for each branch we only need polynomial space. On the other hand, since boolean theoremhood is CONP-Complete, equational reasoning can be encoded in the method, the use of rule **Ref** *is convenient for the presentation of the calculus, but it is not optimal for implementing the decision method. That is, we apply the rest of the rules until we get all the basic deontic formulas and equations, and then we check (for instance) if there are some* $\mathbf{P}(\alpha)$ *and* $\neg\mathbf{P}(\beta)$ *belonging to the branch and check if equation* $\alpha = \beta$ *can be proven from the equations already in the branch, this can be done in PSPACE, and similar for the rest of contradictory formulas.*

6. Final Remarks

In this article we have described tableau methods for deontic action logics based on finite boolean algebras, these logics have been investigated in the literature, but no automated methods of reasoning seem available for them (though as remarked in Section 1, in [6] a tableau calculus is given for DAL[5]). Furthermore, we proved that SAT is NP-Complete for DAL (in any of its versions) when vocabularies are fixed, and the problem is Σ_2^p-Hard when vocabularies are taken as parameters, note that for most modal logics SAT is in PSPACE, the "improvement" in the complexity seems to reside on the simplicity of the semantics of DAL, where no Kripke structures or similar formal devices are used. The proof methods provided here for DAL are in PSPACE, this seems reasonable taking into account the computational complexity of the logic. As already remarked in Section 1, the usefulness of deontic action logics for reasoning about computing systems has been noted by several authors; however, automated proof methods amenable to be implemented in software tools are needed to be able to apply these formalisms in practice; in particular, when one intends to study complex and large computing systems. Thus, we believe that the proof methods introduced in this paper can be used as a basis for constructing (semi) automatic theorem provers for deontic action logics, we leave this as a future work.

References

[1] Åqvist L., Deontic Logic, in D. Gabbay (eds), *Handbook of Philosophical Logic*, vol. 8, Springer Netherlands, 2002.

[2] Anglberger A.J.J., Dynamic Deontic Logic and Its Paradoxes, *Studia Logica*, 89:427–435, 2008.

[3] Arora S., B. Barak, Computational Complexity: A Modern Approach, Cambridge University Press, 2009.

[4] Broersen, J. M., *Modal Action Logics for Reasoning about Reactive Systems,* PhD thesis, Vrije University, 2003.

[5] Broersen, J. M., Action Negation and Alternative Reductions for Dynamic Deontic Logics, *Journal of Applied Logic*, 2: 153–168, 2004.

[6] Castro P.F., T.S.E. Maibaum, A Tableaux System for Deontic Action Logic, *9th International Conference DEON*, *Lecture Notes in Computer Science*, vol. 5076, pp. 34–48, Springer, 2008.

[7] Castro, P.F., T.S.E. Maibaum, Deontic Action Logic, Atomic Boolean Algebras and Fault-Tolerance, *Journal of Applied Logic*, 7(4): 441-466 (2009)

[8] Chellas B.F., *Modal Logic: An Introduction*, Cambridge University Press, 1980.

[9] Dignum, F., J.-J.Ch Meyer, and R.J. Wieringa Free Choice and Contextually Permitted Actions, *Studia Logica*, 57:193–220, 1996

[10] Emerson E.A., Temporal and Modal Logic, in J. van Leeuwen (eds.), *Handbook of Theoretical Computer Science (vol. B)*, MIT Press, 1990.

[11] Fitting, M, *First-Order Logic and Automated Theorem Proving*, Springer-Verlag, 1996.

[12] Harel, D., Dynamic Logic, in D.Gabbay and F.Guenthener (eds.), *Handbook of Philosophical Logic, Volume II: Extensions of Classical Logic*, Reidel, 1984, pp. 497-604.

[13] Hilpinen, R., P. McNamara, Deontic Logic, A Historic Survey and Introduction, In D. Gabbay, J. Horty, R. van der Meyden, X. Parent, L. van der Torre (eds), *Handbook of Deontic Logic and Normative Systems*, College Publications, 2013.

[14] Hughes, J., and L.M.M. Royakkers, Don't ever do that! Long-term duties in PDeL, *Studia Logica*, 89:59–79, 2008.

[15] Kalinowski, J. Theorie des Propositions Normatives, *Studia Logica*, 1:147–182, 1953.

[16] Kent S., W. Quirk, T.S.E. Maibaum, Specifying Deontic Behaviour in Modal Action Logic. *Technical report, Forest Research Project*, 1991.

[17] Knuuttila, S., The Emergence of Deontic Logic in the Fourteenth Century, in R. Hilpinen (eds.), *New Studies in Deontic Logic: Norms, Actions and the Foundations of Ethics*, Springer, 1981, pp. 225–248

[18] Mally, E., *Grungesetza de Sollens: Elemente del Logik des Willens*, Leuscher and Lubensky, Graz, 1926. Reprinted in Mally. E., *Logische Schriften: Großes Logikfragment-Grundgesetze des Sollens*, pp.227–324, edited by K. Walf and P.Weingartner, D. Reidel, Dordrecht, 1971.

[19] Meyer J. J-.Ch., A Different Approach to Deontic Logic: Deontic Logic viewed as a Variant of Dynamic Logic, *Notre Dame Journal of Formal Logic*, 29:106-136, 1988.

[20] Meyer J. J-,Ch. , F.P.M. Dignum, R.J. Wieringa The Paradoxes of Deontic Logic Revisited: A Computer Science Perspective, *Technical Report*, Ultrech University, 1994.

[21] Pratt V., A Practical Decision Method for Propositional Dynamic Logic, in Proceedings of the 10th Annual ACM Symposium on Theory of Computing, New York, NY: ACM, 326?337.

[22] SEGERBERG K., A Deontic Logic of Action, *Studia Logica*, 41:269–282, 1982.

[23] SEGERBERG K, A Blueprint for Dynamic Deontic Logic, *Journal of Applied Logic*, 7:388–402, 2009.

[24] SIKORSKI R., *Boolean Algebras*, Springer-Verlag, 1969.

[25] SMULLYAN R.M, *First-order logic*, Springer-Verlag, 1968.

[26] TRYPUZ R., KULICKI, On Deontic Logics Based in Boolean Algebra, *J. Log. Comput.*, 25:1241-1260, 2015.

[27] VAN DEL MEYDEN, R. The Dynamic Logic of Permission, *Journal of Logic and Computation*, 6:465–479, 1996.

[28] VON WRIGHT, G. H, Deontic Logic, *Mind*, 60: 1-15, 1951.

[29] WIERINGA, R.,J., J.-J.Ch. MEYER, Application of Deontic Logic in Computer Science: A Concise Overview, in R.J.Wieringa, J.-J.Ch. Meyer (eds), *Deontic Logic in Computer Science*, John Wiley & Sons, Inc, 1994.

PABLO F. CASTRO
Departamento de Computación
Universidad Nacional de Rio Cuarto
Ruta 36 Km 601
Rio Cuarto, Argentina
pcastro@dc.exa.unrc.edu.ar