Automated Reasoning over Deontic Action Logics with Finite Vocabularies

Pablo F. Castro
CONICET
Argentina
Departamento de Computación
Universidad Nacional de Río Cuarto
Río Cuarto, Argentina.
pcastro@dc.exa.unrc.edu.ar

T.S.E.Maibaum
Department of Computing & Software
McMaster University
Hamilton (ON), Canada.
tom@maibaum.org

In this paper we investigate further the tableaux system for a deontic action logic we presented in previous work. This tableaux system uses atoms (of a given boolean algebra of action terms) as labels of formulae, this allows us to embrace parallel execution of actions and action complement, two action operators that may present difficulties in their treatment. One of the restrictions of this logic is that it uses vocabularies with a finite number of actions. In this article we prove that this restriction does not affect the coherence of the deduction system; in other words, we prove that the system is complete with respect to language extension. We also study the computational complexity of this extended deductive framework and we prove that the complexity of this system is in PSPACE, which is an improvement with respect to related systems.

1 Introduction

Tableau systems [23, 14, 10] are practical proof systems that are representative of an important stream of research in automated theorem proving [9]. The basic idea behinds these kinds of proof systems is proving by refutation, i.e., to prove a formula φ, we start with $\neg\varphi$ and then we try to derive a contradiction using the rules provided by the logical system. Usually, if a formula is not provable, we get a counterexample (a model which satisfies the negation of the formula). Several tableau systems have been proposed for logics used in computer science, some examples are: *dynamic logics* [21, 13], *modal logics* [10, 18] and *temporal logics* [14, 7].

In [4] we introduced a tableaux method for the deontic action logic presented in [5]; this logic is a *modal action logic* [16] which uses boolean operators on actions (parallel execution and complement of actions) as well as the standard deontic predicates over actions, i.e., *permission*, *obligation* and *prohibition*. We have proposed this logic for reasoning about fault-tolerant programs [5]; the deontic predicates seem suitable to formalize concepts such as *fault*, *violation*, and *fault-recovery*, some intuitions and examples of the use of this logic in specification and verification of fault-tolerant systems are shown in [3]. We believe that tableau methods can help us to provide automated theorem provers for this logic, enabling automation of the analysis of software specifications.

In this paper we extend the system presented in [4] and present further results. In particular, we redefine the rules of the system in such a way that the method for checking formula validity is in PSPACE, and we prove that the restriction to finite action vocabularies does not affect the completeness of the method, this was not proven in the paper cited above. In technical words, we consider a finite number of actions in vocabularies; this implies that extending the vocabularies may affect the validity of formulae, for instance, we may add some extra actions in a vocabulary that may introduce new scenarios in a model

Submitted to:
F

that could falsify a formula that has been proven valid for the original vocabulary. We provide a formal machinery to tackle this problem; corollary 2 of section 4 states that there is a bound on the number of actions to be considered when proving the validity of formulae. More precisely, given a formula, we can calculate the number of actions that we must consider in the vocabulary to prove its "global" validity (i.e., its validity in every vocabulary). Having a finite number of action has some theoretical benefits, for instance, this implies that the underlying algebra of action is atomic, and these atoms can be used as labels of formulae to build canonical models, this also implies that the logic is compact in contrast with similar logics, we discuss this in section 2.1. Furthermore, in section 5 we study the complexity of this tableaux system and we show that the system is in PSPACE, that is, it is aligned with the complexity of most modal logics. Interestingly, most of the dynamic logics with boolean operators proposed in the literature are EXPTIME-Complete.

The paper is organized as follows. In the next section we give a brief description of the deontic action logic. In section 3 we introduce the tableaux system for this logic together with the basic properties of this system. In section 4 we prove some theorems about how formula validity is preserved when the vocabulary is extended. Finally, in section 5 we investigate the complexity of the methods proposed below, and then we discuss some conclusions.

2 Background

Deontic action logics [19, 2, 16] (also called dynamic deontic logics) are modal action logics [16, 15] with deontic predicates over actions. These logics can be classified as "*ought-to-do*" deontic logics, since the deontic operators are applied to actions, in contrast to "*ought-to-be*" logics where deontic operators are applied to predicates; for example, the standard deontic system KD [6] is an "ought-to-be" deontic logic. In this section we describe, briefly, the deontic logic (called DPL) presented in [5]; for further details, the reader is referred to that paper.

The language of the logic is given by a vocabulary $V = \langle \Delta_0, \Phi_0 \rangle$, where Δ_0 is a finite set of primitive actions (denoted by $a, b, c, d, \ldots$), and a set Φ_0 of primitive propositions (denoted by $p, q, s, \ldots$). Using these two sets one can build complex formulae by employing the modal connectives, the deontic predicates and the boolean operators over actions. The set Δ of action terms and the set Φ of formulae over V are described by the following grammars:

$$\alpha ::= a_i \mid \alpha \sqcap \alpha \mid \alpha \sqcup \alpha \mid \overline{\alpha} \mid \emptyset \mid \mathbf{U}$$
$$\varphi ::= p_i \mid \neg\varphi \mid \varphi \rightarrow \varphi \mid \mathsf{P}(\alpha) \mid \mathsf{P_w}(\alpha) \mid [\alpha]\varphi \mid \alpha_1 =_{act} \alpha_2$$

where $a_i \in \Delta_0$. $\alpha_1 \sqcup \alpha_2$ is the non-deterministic choice between actions α_1 and α_2, $\alpha_1 \sqcap \alpha_2$ is the parallel execution of actions α_1 and α_2, $\overline{\alpha}$ denotes the execution of an alternative action to α, $\emptyset$ is an impossible action and $\mathbf{U}$ denotes the execution of any action. In addition to the standard boolean connectives, we have the following formulae: $[\alpha]\varphi$ asserts that after any execution of α, φ is true; and $\alpha_1 =_{act} \alpha_2$ states that actions α_1 and α_2 are equal. We consider two permission predicates: $\mathsf{P_w}(\alpha)$ is called *weak* permission; it is true when α is allowed to be executed in some scenarios. On the other hand, $\mathsf{P}(\alpha)$ is called *strong* permission; this formula is true when α is permitted to be executed in any scenario. The two versions of permission can be found in the deontic literature. Using permissions, we introduce other deontic operators such as obligation, prohibition, etc. Note that in this logic the interpretation of deontic predicates is independent of the modal operators (they have a different interpretation in the semantic structures), whereas in related work (e.g., [19, 2]) the deontic operators are reduced to modal formulae.

Let us briefly introduce the semantics of the logic with some remarks:

Definition 1 (structures) *Given a vocabulary $V = \langle \Phi_0, \Delta_0 \rangle$, a V-Structure is a tuple $M = \langle \mathscr{W}, \mathscr{R}, \mathscr{E}, \mathscr{I}, \mathscr{P} \rangle$ where:*

- *$\mathscr{W}$ is a set of worlds;*
- *$\mathscr{R}$ is an $\mathscr{E}$-labeled relation between worlds, s.t., if $(w, w', e) \in \mathscr{R}$ and $(w, w'', e) \in \mathscr{R}$, then $w' = w''$;*
- *$\mathscr{E}$ is a non-empty set of (names of) events.*
- *$\mathscr{P} \subseteq \mathscr{W} \times \mathscr{E}$ is a relation which indicates which event is permitted in which world;*
- *$\mathscr{I}$ is a function s.t. for every $p \in \Phi_0 : \mathscr{I}(p) \subseteq \mathscr{W}$ and for every $\alpha \in \Delta_0 : \mathscr{I}(\alpha) \subseteq \mathscr{E}$.*

$\mathscr{I}$ has to satisfy the following properties:

I.1 *For every $\alpha_i \in \Delta_0$: $|\mathscr{I}(\alpha_i) - \bigcup\{\mathscr{I}(\alpha_j) \mid \alpha_j \in (\Delta_0 - \{\alpha_i\})\}| \leq 1$.*

I.2 *For every $e \in \mathscr{E}$: if $e \in \mathscr{I}(\alpha_i) \cap \mathscr{I}(\alpha_j)$, where $\alpha_i \neq \alpha_j \in \Delta_0$, then:*
$\cap\{\mathscr{I}(\alpha_k) \mid \alpha_k \in \Delta_0 \wedge e \in \mathscr{I}(\alpha_k)\} = \{e\}$.

I.3 $\mathscr{E} = \bigcup_{\alpha_i \in \Delta_0} \mathscr{I}(\alpha_i)$.

We can extend the function $\mathscr{I}$ to well-formed formulae and action terms as follows:

- $\mathscr{I}(\neg\varphi) \stackrel{\text{def}}{=} \mathscr{W} - \mathscr{I}(\varphi)$,
- $\mathscr{I}(\varphi \rightarrow \psi) \stackrel{\text{def}}{=} \mathscr{I}(\neg\varphi) \cup \mathscr{I}(\psi)$,
- $\mathscr{I}(\alpha \sqcup \beta) \stackrel{\text{def}}{=} \mathscr{I}(\alpha) \cup \mathscr{I}(\beta)$,
- $\mathscr{I}(\mathbf{U}) \stackrel{\text{def}}{=} \mathscr{E}$,
- $\mathscr{I}(\alpha \sqcap \beta) \stackrel{\text{def}}{=} \mathscr{I}(\alpha) \cap \mathscr{I}(\beta)$,
- $\mathscr{I}(\overline{\alpha}) \stackrel{\text{def}}{=} \mathscr{E} - I(\alpha)$,
- $\mathscr{I}(\emptyset) \stackrel{\text{def}}{=} \emptyset$.

Note that here we do not follow the traditional approach of interpreting each action as a relation (e.g., see [12]); instead we interpret each action as a set of "*events*", the events in which it "*participates in*" during its execution. Then, the action combinators are interpreted as the classical boolean set operators. Note that the restrictions on models (**I.1** and **I.2**) imply that we have one point sets in the family of the event sets, i.e., intuitively every "event" is produced by a combination of actions in our systems (system actions and environmental actions). (So, events can be seen as collections of actions to be executed in parallel.) Then, if we take a maximal set of actions, the execution of this set only produces an event in our system; in other words, this set of actions is complete in the sense that they describe unambiguously one event in the system execution. We have presented a sound and complete axiomatic system for this logic in [5]. Given a structure $M = \langle \mathscr{W}, \mathscr{R}, \mathscr{E}, \mathscr{I}, \mathscr{P} \rangle$, we define the relation $\vDash$ between worlds, models, and formulae, is as follows:

- $w, M \vDash p \stackrel{\text{def}}{\Longleftrightarrow} w \in \mathscr{I}(p)$.
- $w, M \vDash \varphi \rightarrow \psi \stackrel{\text{def}}{\Longleftrightarrow}$ not $w, M \vDash \varphi$ or $w, M \vDash \psi$.
- $w, M \vDash \neg\varphi \stackrel{\text{def}}{\Longleftrightarrow}$ not $w, M \vDash \varphi$.
- $w, M \vDash \alpha =_{act} \beta \stackrel{\text{def}}{\Longleftrightarrow} \mathscr{I}(\alpha) = \mathscr{I}(\beta)$.

- $w, M \vDash [\alpha]\varphi \overset{\mathsf{def}}{\Longleftrightarrow}$ for all $w' \in \mathscr{W}$ and $e \in \mathscr{I}(\alpha)$ if $w \xrightarrow{e} w'$ then $w', M \vDash \varphi$.
- $w, M \vDash \mathsf{P}(\alpha) \overset{\mathsf{def}}{\Longleftrightarrow}$ for all $e \in \mathscr{I}(\alpha)$, $\mathscr{P}(w, e)$ holds.
- $w, M \vDash \mathsf{P_w}(\alpha) \overset{\mathsf{def}}{\Longleftrightarrow}$ there exists some $e \in \mathscr{I}(\alpha)$ such that $\mathscr{P}(w, e)$

For the other standard formulae the definition is as usual.

2.1 Related Logics

We make a brief digression to compare the logic presented above with related formalisms. Boolean Modal Logic [12, 1] combines modal operators with boolean operators over actions. In this logic a relational semantics is provided: each action is interpreted as a relationship in the semantic structures. In this setting, the universal action is interpreted as the universal relationship between states or worlds. In the logic presented above, the universal action is relative to the actual state, that is, this action characterizes all the reachable states from a given state, and therefore the complement is also relative to the actual state. As it is argued in [2], relative complements are more useful when reasoning about computer systems. To the authors' knowledge, no tableau systems have been proposed for boolean modal logic with relative complement.

It is worth remarking that in our logic the existence of boolean atoms in the boolean algebra of actions allows us to express exactly which actions are involved in a given transition: each atomic action denotes exactly one event. This allows us to prove the strong completeness of the logic and therefore we get also the compactness of the axiomatic system presented in [5]. BML is not compact; this can be easily proven using the fact that the vocabularies in BML contain an infinite number of actions [2].

On the other hand, Segerberg [22] presents a deontic action logic with boolean operators where permissions are characterized as ideals of the boolean algebra of actions. As shown in [24], the absence of atoms in Segerberg's logic implies that the so-called closure principle (*any event is allowed or forbidden*) cannot be captured by Segerberg's logic. In section 5 we compare the time complexity of the tableaux method proposed below with the complexity of the logics referenced in this section.

3 A Tableaux System for DPL

In this section we describe the tableaux system introduced in [4], we introduce some minor changes to the original system to be able to improve its complexity (see below); note that formulae are enriched with labels; intuitively, each label indicates a state in the semantics where the formula is true. Labeled systems are usual for many logics and tableaux systems, an introduction to these systems can be found in [10, 11]. We adapt these techniques to our modal action logic, showing that deontic operators fit neatly into the system; the duality between the strong and weak permissions resembles the duality between modal necessity and modal possibility. As shown in [4] this system is sound and complete.

A labeled, or prefixed, formula has the following structure: $\sigma : \varphi$, where σ is a label made up of a sequence of boolean (action) terms built from a given vocabulary. We use the following notation for sequences: $\langle\rangle$ (*the empty sequence*), $x \,\boldsymbol{.}\, xs$ (*the sequence made of an element x followed by a sequence xs*); we also use the same notation to denote the concatenation of two sequences.

From here on we consider a fixed vocabulary: $V = \langle \Phi_0, \Delta_0 \rangle$. We denote by Φ_{BA} some complete and decidable axiomatization of boolean algebras [20]; If $\Delta_0 = \{a_1, ..., a_n\}$, we add the equation $a_1 \sqcup \cdots \sqcup a_n = \mathbf{U}$ to the set Φ_{BA}. We denote by Δ_0/Γ the boolean terms over Δ_0 modulo a set of axioms Γ; usually, Γ is an extension of the theory of boolean algebras, i.e., $\Phi_{BA} \subseteq \Gamma$. We write $\Gamma \vdash_{BA} t_1 =_{act} t_2$, if the equation $t_1 =_{act} t_2$ is provable from the boolean theory Γ using equational calculus. This implies that our method depends on some suitable method to decide boolean algebras. Using this notation, we denote by $At(\Delta_0/\Gamma)$ the set of atoms in the boolean algebra of terms modulo $\Gamma \cup \Phi_{BA}$ (note that the boolean algebra is atomic because the set of primitive action symbols is finite). In the same way, we denote by $At_{\sqsubseteq\alpha}(\Delta_0/\Gamma)$ the set of atoms $\gamma \in At(\Delta_0/\Gamma)$ such that $\Gamma \vdash_{BA} \gamma \sqsubseteq \alpha$, where $\sqsubseteq$ is the order relation of the algebra, and $At_{\sqsubset\alpha}(\Delta_0/\Gamma)$ denotes the strict version of this set.

A *tableau* is an (n-ary) rooted tree where nodes are labeled with prefixed formulae, and a *branch* is a path from the root to some leaf. Intuitively, a branch is a tentative model for the initial formula (which we are trying to prove valid). Given a branch $\mathscr{B}$, we denote by $EQ(\mathscr{B})$ the equations appearing in $\mathscr{B}$.

In figure 1 we introduce a classification of formulae which is useful for presenting the rules of the tableaux calculus (see figure 2). Standard propositional formulae are classified following Smullyan's unifying notation [23]. We also introduce the less standard classification for modal logics. (We follow the standard notation for modal logics [10].) For each prefixed formula of type P or N, we define formulae $P(\gamma)$ and $N(\gamma)$, respectively. Here γ is some action term which is needed to define these formulae (see the rules below). Note that for any formula P, $P(\gamma)$ denotes two formulae. Finally, we introduce a new classification for deontic formulae (formulae P_D and N_D). Although the deontic operators are, in some sense, similar to the modal operators, we need to distinguish them; the deontic predicates state properties about transitions, whereas the modal operators state properties about states related to the actual state. Using the above classification of formulae, we can introduce the rules of the tableaux method. In the definition of these rules we use *front action* of a P, N, P_D or N_D formula to refer to the action nearest the root in the syntax tree corresponding to this formula.

A	A_1	A_2
$\sigma : \varphi \wedge \psi$	$\sigma : \varphi$	$\sigma : \psi$
$\sigma : \neg(\varphi \vee \psi)$	$\sigma : \neg\varphi$	$\sigma : \neg\psi$
$\sigma : \neg\neg\varphi$	$\sigma : \varphi$	

P	$P(\gamma)$
$\sigma : \langle\alpha\rangle\varphi$	$\sigma \bullet \gamma : \varphi,\ \sigma : \gamma \neq_{act} \emptyset$
$\sigma : \neg[\alpha]\varphi$	$\sigma \bullet \gamma : \neg\varphi,\ \sigma : \gamma \neq_{act} \emptyset$

N	$N(\gamma)$
$\sigma : [\alpha]\varphi$	$\sigma \bullet \gamma : \varphi$
$\sigma : \neg\langle\alpha\rangle\varphi$	$\sigma \bullet \gamma : \neg\varphi$

B	B_1	B_2
$\sigma : \varphi \vee \psi$	$\sigma : \varphi$	$\sigma : \psi$
$\sigma : \neg(\varphi \wedge \psi)$	$\sigma : \neg\varphi$	$\sigma : \neg\psi$

P_D	$P_D(\gamma)$
$\sigma : \neg\mathsf{P}(\alpha)$	$\sigma : \neg\mathsf{P}(\gamma),\ \sigma : \gamma \neq_{act} \emptyset$
$\sigma : \mathsf{P_w}(\alpha)$	$\sigma : \mathsf{P_w}(\gamma),\ \sigma : \gamma \neq_{act} \emptyset$

N_D	$N_D(\gamma)$
$\sigma : \mathsf{P}(\alpha)$	$\sigma : \mathsf{P}(\gamma)$
$\sigma : \neg\mathsf{P_w}(\alpha)$	$\sigma : \neg\mathsf{P_w}(\gamma)$

Figure 1: Classification for deontic formulae.

The rules of this calculus can be found in figure 2. The rules for standard boolean operators are as usual. Rule N is standard for **K** modal logics [10]; it does not introduce new labels in the branch, but it adds new formulae to labels already in the branch; intuitively, for all (the states denoted by) the labels reachable from the current state, the N formula must be true. The rule for deontic necessity is similar, but it adds the corresponding deontic formulae to labels already in the branch and for which there is a P_D formula with the same action in the branch.

Notice the rules P and P_D for modal and deontic possibility, respectively; given a P formula, rule P creates one branch for each possible execution of the front action in the formula; although the rule for deontic possibility is very similar, note that deontic possibility does not create new labels, because permissions only predicate over transitions. Note that, in these rules, an inequation saying that the action must not be impossible is added in each branch, allowing us to avoid adding labels that cannot exist in the semantics. Finally, rule *Per* states that, if an action which is atomic (in the sense that it cannot have different executions, i.e., not participate in different events) is weakly allowed, then it is also strongly

$$P\colon \quad \frac{P}{P(\gamma_1) \mid ... \mid P(\gamma_n)}$$

with $\{\gamma_1,\ldots,\gamma_n\} = At_{\sqsubset\alpha}(\Delta_0/\mathit{\Gamma})$, α is the front action of P and $\mathit{\Gamma}$ is the set of equations in the branch.

$$P_D\colon \quad \frac{P_D}{P_D(\gamma_1) \mid ... \mid P_D(\gamma_n)}$$

with $\{\gamma_1,\ldots,\gamma_n\} = At_{\sqsubset\alpha}(\Delta_0/\mathit{\Gamma})$, α is the front action of P_D and $\mathit{\Gamma}$ is the set of equations in the branch.

$$A : \frac{A}{\begin{array}{c} A_1 \\ A_2 \end{array}}$$

$$B : \frac{B}{B_1 \mid B_2}$$

$$Per : \frac{\sigma : \mathsf{P_w}(\gamma)}{\sigma : \mathsf{P}(\gamma)}$$

$$N_D : \quad \frac{N_D}{\begin{array}{c} N_D(\gamma_1) \\ \vdots \\ N_D(\gamma_n) \end{array}}$$

for all $\gamma_1, ..., \gamma_n \in At_{\sqsubset\alpha}(\Delta_0/\Gamma)$, for α the front action of N_D, Γ the set of equations appearing in the branch and such that for each γ_i there is already a $P_D(\gamma_i)$ formula with the same label to N_D in the branch.

$$N : \quad \frac{N}{\begin{array}{c} N(\gamma_1) \\ \vdots \\ N(\gamma_n) \end{array}}$$

for all $\gamma_1, ..., \gamma_n \in At_{\sqsubset\alpha}(\Delta_0/\Gamma)$, for α the front action of N, Γ the set of equations appearing in the branch and where the labels of $N(\gamma_i)$ are already in the branch.

Figure 2: Tableau Rules

allowed.

We do not state any rule for equality; this is because equality reasoning is implicit in our calculus (see below the definition of boolean closed). For simplicity of the presentation of the concepts, we rule out those formulae of the form: $[\alpha](\alpha =_{act} \beta)$. This does not affect the completeness of the method since formulae of this kind are equivalent to formulae where equations do not appear after modalities [5]. Let us introduce the notions of *closed*, *boolean closed*, *deontic closed* and *open branch*. Keep in mind that a branch is a set of prefixed formulae.

Given a branch $\mathscr{B}$ and a boolean theory Γ, we say that $\mathscr{B}$ is *deontic closed* with respect to Γ if it satisfies at least one of the following conditions: (i) $\sigma : \mathsf{P}(\alpha) \in \mathscr{B}$ and $\sigma : \neg\mathsf{P}(\alpha) \in \mathscr{B}$, for some label σ; (ii) $\sigma : \mathsf{P_w}(\alpha) \in \mathscr{B}$ and $\sigma : \neg\mathsf{P_w}(\alpha) \in \mathscr{B}$, for some label σ; (iii) $\sigma : \neg\mathsf{P}(\alpha) \in \mathscr{B}$ and $\sigma : \mathsf{P_w}(\alpha) \in \mathscr{B}$, for some label σ.

Note that we have not included $\sigma : \mathsf{P}(\alpha)$ and $\sigma : \neg\mathsf{P_w}(\alpha)$ as being mutually contradictory; this is because they are not contradictory when $\Gamma \vdash_{BA} \alpha =_{act} \emptyset$. This fact yields the definition of *extended boolean theory*:

$$EQ^*(\mathscr{B}) = \{(\alpha \sqcap \beta =_{act} \emptyset) \mid \sigma : \mathsf{P}(\alpha), \sigma : \neg\mathsf{P_w}(\beta) \in \mathscr{B}\} \cup EQ(\mathscr{B})$$

It is useful for us to introduce the notion of *boolean closed* branch; intuitively these branches are inconsistent boolean theories. A branch $\mathscr{B}$ is *boolean closed* iff $EQ^*(\mathscr{B}) \vdash_{BA} \emptyset =_{act} \mathbf{U}$, or $EQ^*(\mathscr{B}) \vdash_{BA} \alpha =_{act} \beta$ and $\alpha \neq_{act} \beta \in \mathscr{B}$

Finally, we say that a branch is *closed* if either it contains a labeled propositional variable $\sigma : p$ and a labeled negation of it $\sigma : \neg p$, or it is deontic closed or boolean closed. Note that rule N_D only takes into account labels that contain a P_D formula, in [4] this rule creates a labelled deontic formula for each atom of the corresponding action, implying that in that system the space needed in a branch is exponential w.r.t. formula length.

4 Completeness with respect to language extension

Recall that vocabularies contain a finite number of primitive actions. This is different from what happens in dynamic logics, where vocabularies have an infinite number of actions. This assumption allows us to obtain atoms in the corresponding boolean algebra of action terms. These atoms are useful for proving completeness and compactness. However, doing this we also constrain our deductive machinery to only take into account a restricted number of actions. Sometimes, we will be interested in proving properties that are valid in every vocabulary, and not only in a particular one. This extension idea then represents situations where we have embedded our component in a larger one with a set of primitive actions extending those of the component.

Consider the formula: $\langle\alpha\rangle\varphi \rightarrow [\alpha]\varphi$, this formula is not valid, but if we build the tableau for it considering a vocabulary with a as the unique action, the final tree has no open branches. This only shows that this formula is valid for a vocabulary with one primitive action. Intuitively, if we think of a theory as a specification of a computing system, we take the view that the vocabulary describes all the actions that can be executed during a running of the system being specified. Adding more actions to the vocabulary can be understood as incorporating new behavior to the system, or taking into account more actions from the environment (e.g., some additional interaction with the users). Sometimes, we will be interested in proving that some properties are valid in any vocabulary, this has the obvious interpretation that these are properties that are valid for a system and any extension of it.

Summarizing, our specification only gives us a partial picture of a system. Because of this, system properties are hard to verify (using tableaux or other formal systems). After all, perhaps we may not be taking into account some actions important for the property to be proven. The following theorems give us some machinery to address this difficulty. Corollary 2 says that we can verify a property (using tableaux) restricting our attention to a finite number of actions; if for this number of actions, this formula is valid, then it will be valid for any language extension (containing, potentially, any number of actions). Some auxiliary notions are needed and we introduce the concepts of *normal form*, *disjunctive normal form*, and *existential degree*, and then we present the theorems.

The *degree* of a formula φ (denoted by $\mathsf{d}(\varphi)$) is the length of the longest string of nested modalities (taking permission as being of degree 0). For any formula φ we denote by $Pr(\varphi)$ the set of primitive actions appearing in φ. Given a vocabulary $\langle\Delta_0, \Phi_0\rangle$, we adapt the definition of *normal form of degree n* given in [8] to our logic. We denote by F_i the set of formulae of normal form of degree i, defined as follows:

- F_0 is the set of formulae of the form $*\varphi_1 \wedge ... \wedge *\varphi_h$, where for each i: $\varphi_i \in \Phi_0$ or φ_i is a deontic predicate, and $*$ is $\neg$ or blank.
- F_{n+1} is the set of formulae of the form: $\theta \wedge *\langle\alpha_1\rangle\varphi_1 \wedge .... \wedge *\langle\alpha_k\rangle\varphi_k$, where $\theta \in F_0$, $\varphi_i \in F_n$ for all $1 \leq i \leq k$, $*$ is $\neg$ or blank. (θ may not appear in the formula, in which case we only consider everything but not θ.)

The set of normal form formulae is $F = \bigcup_{i=1}^{\infty} F_i$. If a formula is in normal form, we say that it is a NF formula. Any formula of degree $\leq n$ is equivalent to $\perp$ or a disjunction of normal forms of degree n:

Theorem 1 *Given a vocabulary V and formula φ of degree $\leq n$, then either there exist NF formulae ϕ_i of degree n, such that $w, M \vDash \varphi \leftrightarrow \bigvee \varphi_i$ for any V'-structure M (with $V \subseteq V'$); or there is no V'-structure M s.t. $w, M \vDash \varphi$*

Proof *See the proof given in [8] and use the property $\vdash \langle\alpha\rangle(\varphi \vee \psi) \leftrightarrow \langle\alpha\rangle\varphi \vee \langle\alpha\rangle\psi$, which is valid in any vocabulary.*

Note that, in general, a NF formula can be expressed using the following schema:

$$\theta \wedge \Delta \wedge \bigwedge_{i=1}^{n} \langle \alpha_i \rangle \varphi \wedge \bigwedge_{j=1}^{m} \neg \langle \beta_j \rangle \psi$$

where θ is a conjunction of propositional variables or negations of them, and Δ is a conjunction of deontic predicates or negations of them.

If a formula is a disjunction of normal forms, we say that this formula is in *disjunctive normal form* (or DNF for short). We call $\mathsf{P}_{\mathsf{w}}(\alpha)$, $\neg\mathsf{P}(\alpha)$ and $\langle \alpha \rangle \varphi$ *existential formulae*, i.e., *existential formulae* are those whose semantics is given in terms of an existential quantifier. Given a NF formula φ, with $\mathsf{d}(\varphi) = n$, we can define a set of formulae $\mathsf{SF}(\varphi, k)$, for every $k \leq n$, called the *subformulae at level k*. For $k = 0$ we define:

- If $\varphi = \bigwedge_{i=1}^{n} *p_i$, i.e., it is a conjunction of propositions or negations of them, then for this case the definition is: $\mathsf{SF}(\bigwedge_{i=1}^{n} *p_i, 0) = \bigcup_{i=1}^{n} \{*p_i\}$
- If $\varphi = \bigwedge_{j=1}^{m} *\mathsf{P}(\alpha_j) \wedge \bigwedge_{k=1}^{t} *\mathsf{P}_{\mathsf{w}}(\beta_k)$, i.e., the formula is a conjunction of deontic formulae or negations of them, then we define:

$$\mathsf{SF}(\bigwedge_{j=1}^{m} *\mathsf{P}(\alpha_j) \wedge \bigwedge_{k=1}^{t} *\mathsf{P}_{\mathsf{w}}(\beta_k), 0) = \bigcup_{j=1}^{m} \{*\mathsf{P}(\alpha_j)\} \cup \bigcup_{k=1}^{t} \{*\mathsf{P}_{\mathsf{w}}(\beta_k)\}$$

- In the case of a conjunction of propositional formulae and deontic formulae we can use the two definitions above, that is: $\mathsf{SF}(\theta \wedge \Delta, 0) = \mathsf{SF}(\theta, 0) \cup \mathsf{SF}(\Delta, 0)$
- In the general case, we define:

$$\mathsf{SF}(\theta \wedge \Delta \wedge (\bigwedge_{i=1}^{n} \langle \alpha_i \rangle \varphi_i) \wedge (\bigwedge_{j=q}^{m} \neg \langle \beta_j \rangle \psi_j), 0) = \mathsf{SF}(\theta) \cup \mathsf{SF}(\Delta) \cup \bigcup_{i=1}^{n} \{\langle \alpha_i \rangle \varphi_i\} \cup \bigcup_{j=1}^{m} \{\neg \langle \beta_j \rangle \psi_j\}$$

For the case of $k > 0$, we define:

$$\mathsf{SF}(\theta \wedge \Delta \wedge (\bigwedge_{i=1}^{n} \langle \alpha_i \rangle \varphi_i) \wedge (\bigwedge_{j=q}^{m} \neg \langle \beta_j \rangle \psi_j), k+1) = \bigcup_{i=1}^{n} \mathsf{SF}(\varphi_i, k) \cup \bigcup_{j=1}^{m} \neg \mathsf{SF}(\psi_j, k)$$

where given a set S of formulae, we denote by $\neg S$, the set containing the negations of the formulae in S. (We also suppose that several negations over a formula are simplified, i.e., instead of having $\neg\neg p$ we have p.) In some sense, the set SF indicates which set of subformulae must be true at a given level. We use $\#_{\exists} S$ to denote the number of existential formulae in the set S. Using this definition, we can define the *existential degree* of a NF formula φ, denoted by $\mathsf{D}_{\exists}$, which is defined as follows: $\mathsf{D}_{\exists}(\varphi) = \mathsf{max}_{0 \leq i \leq n} \{\#_{\exists} \mathsf{SF}(\varphi, i)\}$. We can extend this definition to DNF formulae, as follows: $\mathsf{D}_{\exists}(\varphi_1 \vee ... \vee \varphi_k) = max\{\mathsf{D}_{\exists}(\varphi_1), .., \mathsf{D}_{\exists}(\varphi_k)\}$

Note that function $\mathsf{D}_{\exists}$ can be extended to cope with any formula: to obtain $\mathsf{D}_{\exists}(\varphi)$ (where φ may not be a DNF formula) calculate the maximum number of existential subformulae in the same level of the syntax tree of such a formula. Note that this number coincides with the existential degree of an equivalent DNF formulae (obtained by using theorem 1).

The idea is to use the sets SF to define smaller models of φ. First, we need to define the notion of n-reachable. Given a model M and a state w, we say that a state v is n-reachable (or reachable in n-steps) from w, if there exists a path $w \xrightarrow{e_1} w_2 \xrightarrow{e_2} \dots \xrightarrow{e_n} v$ in M. Note that our logic has the *unraveling property* [1], i.e., if a formula φ is satisfiable in a model M and state w, we can build a model M' unraveling M such that w and M' satisfies φ and this new model is a tree, i.e., it does not have cycles. For the following results we restrict our attention to tree models; the unraveling property guarantees that these theorems extend to any other model.

If $\mathsf{d}(\varphi) = n$, then we can define a mapping L_w from the states reachable in M from w in n or less steps, to the subformulae of φ, as follows:

$$L_w(v) = \{\psi \in \mathsf{SF}(\varphi, k) \mid v \text{ is } k\text{-reachable from } w \text{ and } v, M \vDash \psi \text{ with } k \leq n\}.$$

Using the definition of $L_w(v)$ we prove that, given a formula φ and a model of this formula, we can define a new model that has an out-degree (the number of transitions coming out of any state) less than or equal to $\mathsf{D}_{\exists}(\varphi)$.

Theorem 2 *Given a NF formula φ and a model $M = \langle \mathscr{W}, \mathscr{R}, \mathscr{E}, \mathscr{I}, \mathscr{P} \rangle$ over a vocabulary $V = \langle \Delta_0, \Phi_0 \rangle$, if $w, M \vDash \varphi$, then there exists a model M^{φ}_{w} such that $v, M^{\varphi}_{w} \vDash L_w(v)$, for every $v \in \mathscr{W}^{\varphi}_{w}$ and M^{φ}_{w} has an out-degree less than or equal to $\mathsf{D}_{\exists}(\varphi)$.*

Proof: *First let us define the new model, consider a model $M = \langle \mathscr{W}, \mathscr{R}, \mathscr{E}, \mathscr{I}, \mathscr{P} \rangle$ over a vocabulary $V = \langle \Delta_0, \Phi_0 \rangle$ and a NF formula φ (of degree n) such that $w, M \vDash \varphi$. The labeling L helps us to define a new model $M^{\varphi}_{w} = \langle \mathscr{W}^{\varphi}_{w}, \mathscr{R}^{\varphi}_{w}, \mathscr{E}^{\varphi}_{w}, \mathscr{I}^{\varphi}_{w}, \mathscr{P}^{\varphi}_{w} \rangle$ as follows:*

- $\mathscr{E}^{\varphi}_{w} = \mathscr{E}$.
- *We define $\mathscr{R}^{\varphi}_{w}$ in n steps:*
  - *At step 0, choose for each $\langle \alpha_i \rangle \varphi_i \in L(w)$ an event e_i such that $e_i \in \mathscr{I}(\alpha_i)$ and there exists a state v_i with $w \xrightarrow{e_i} v_i$, and $v_i, M \vDash \varphi_i$, and define $\mathscr{R}^0 = \bigcup\limits_{e_i} \{w \xrightarrow{e_i} v_i\}$.*
  - *At step $k+1$, let $v_1, ..., v_m$ be the states k-reachable from w. For each of these states proceed as was done for state w, and define a relation $R^k_{v_i}$, and then $\mathscr{R}^{k+1} = \bigcup\limits_{i \leq m} R^k_{v_i}$*

  Finally, $R^{\varphi}_{w} = \bigcup_{k \leq n} R^k$.
- $\mathscr{P}^{\varphi}_{w} = \mathscr{P}$.
- $\mathscr{W}^{\varphi}_{w} = \{v \in \mathscr{W} \mid v \in \mathit{Dom}(\mathscr{R}^{\varphi}_{w}) \cup \mathit{Ran}(\mathscr{R}^{\varphi}_{w})\}$.
- $I^{\varphi}_{w}(a_i) = \mathscr{I}(a_i)$, *for every e_i.* $\mathscr{I}^{\varphi}_{w}(p_i) = \mathscr{I}(p_i)$, *for every $p_i \in \Phi_0$.*

Note that this model has an out-degree (the number of transitions coming out of any state) less than or equal to $\mathsf{D}_{\exists}(\varphi)$, since in each state of the new model, we only have one transition per existential subformula at the corresponding level.

Suppose that $\mathsf{d}(\varphi) = n$; we prove the result by induction. If v is reachable in n steps from w, then, by definition, $L(v)$ only contains propositional variables and deontic predicates, and therefore, by definition of M^{φ}_{w}, we have that $v, M^{\varphi}_{w} \vDash L(v)$.

If v is reachable in k steps (with $k < n$) from w, then for each $\langle \alpha_i \rangle \varphi_i$ in $L(v)$ we have an $e_i \in \mathscr{I}^{\varphi}_{w}(\alpha_i)$ such that $v \xrightarrow{e_i} v'$; by induction we know that $v', M^{\varphi}_{w} \vDash \varphi_i$, and therefore $v, M^{\varphi}_{w} \vDash \langle \varphi_i \rangle \varphi_i$.

Now, suppose that $\neg \langle \beta_j \rangle \psi_j \in L(v)$; we know that ψ_j is a NF formula, and therefore $\psi_j = \psi'_1 \wedge ... \wedge \psi'_h$, and by definition of SF, *we have that $\neg \psi'_1, ..., \neg \psi'_h \in \mathsf{SF}(\varphi, k+1)$. Then, if for some ψ'_l and state v', we have $v', M \vDash \neg \psi'_l$ (where $v \xrightarrow{e_j} v'$ in $\mathscr{R}$ and $e_j \in \mathscr{I}(\beta_j)$), i.e., we have that $\neg \psi'_l \in L(v')$, then by induction we have $v', M^{\varphi}_{w} \vDash \neg \psi'_l$, which implies that $v, M^{\varphi}_{w} \vDash \neg \langle \beta_j \rangle \psi_j$. This concludes the proof.*

Note that $\bigwedge L_w(w) = \varphi$, and therefore we obtain the following corollary.

Corollary 1 *If $w, M \vDash \varphi$, then $w, M^{\varphi}_{w} \vDash \varphi$.*

Summarizing, given a model of a NF formula φ, we can build a new model with branching being at most $\mathsf{D}_{\exists}(\varphi)$. We are close to our original goal; using the model M^{φ}_{w}, we define another model over a restricted vocabulary that preserves property φ.

Theorem 3 *Given a vocabulary $V = \langle \Delta_0, \Phi_0 \rangle$, a NF formula φ such that $\Delta_0 > Pr(\varphi) + \mathsf{D}_\exists(\varphi)$, with $\mathsf{d}(\varphi) = n$ and a model M, if $w, M \vDash \varphi$, then there is a model M^* over a vocabulary $V^* = \langle Pr(\varphi) \cup \{b_1, ..., b_{\mathsf{D}_\exists(\varphi)}\}, \Phi_0 \rangle$ with the b_i's being fresh action terms, such that $w, M^* \vDash \varphi$.*
Proof: *First, given a model M over a vocabulary $V = \langle \Delta_0, \Phi_0 \rangle$, we denote by $EQ(M)$ the set of equations true in M, and if we have a subset $S \subseteq \Delta_0$, we denote by $EQ^S(M)$ the set of equations built from primitive actions in S which are true in M, i.e.,*

$$EQ^S(M) = \{\alpha =_{act} \beta \mid \alpha =_{act} \beta \in EQ(M) \wedge \alpha, \beta \in T_{BA}(S)\}$$

where $T_{BA}(S)$ denotes the set of boolean terms built from variables in S.

Suppose that $\mathsf{D}_\exists(\varphi) = c$. If $\#\Delta_0 > Pr(\varphi) + c$, then we define a model $M^* = \langle \mathscr{W}^*, \mathscr{R}^*, \mathscr{E}^*, \mathscr{I}^*, \mathscr{P}^* \rangle$ over the vocabulary $V^* = \langle \Delta_0^* = Pr(\varphi) \cup \{b_1, \ldots, b_c\}, \Phi_0 \rangle$, $b_1, ..., b_c$ being fresh primitive actions.

- $\mathscr{E}^* = At(\Delta_0^* / EQ^{Pr(\varphi)}(M))$.
- $\mathscr{W}^* = W_w^\varphi$.
- *For each $v \in W_w^\varphi$ let $\{e_1^v, ..., e_k^v\} \subseteq \mathscr{E}_w^\varphi$ be the set of events such that each e_i^v satisfies either:*
  - *there exists a state v_i and $v \xrightarrow{e_i} v_i \in \mathscr{R}_w^\varphi$, or*
  - *there is a $\mathsf{P}_\mathsf{w}(\alpha_i) \in L(v)$ such that $e_i^v \in \mathscr{I}_w^\varphi(\alpha_i)$ and $\mathscr{P}_w^\varphi(v, e_i^v)$, or*
  - *there is a $\neg\mathsf{P}(\alpha_i) \in L(v)$ with $e_i^v \in \mathscr{I}_w^\varphi(\alpha_i)$ and $(v, e_i^v) \notin \mathscr{P}_w^\varphi$*

  We know that $k \leq \mathsf{D}_\exists(\varphi)$, and then define for each such a e_w^v a corresponding event in $\mathscr{E}^*$ as follows:

  $$e_i^{v*} = (\bigsqcap_{a \in Pr(\varphi) \wedge e_i^v \in \mathscr{I}(a)} a) \sqcap (\bigsqcap_{a' \in Pr(\varphi) \wedge e_i^v \notin \mathscr{I}(a')} \overline{a'}) \sqcap (\bigsqcap_{b_j \in \Delta_0' \wedge b_j \neq b_i} \overline{b_j}) \sqcap b_i$$

  (where we use some enumeration of the fresh b's to determine each b_i); note that for these e_i^{v*}'s, we have: $e_i^v \in \mathscr{I}_w^\varphi(\alpha) \Leftrightarrow e_i^{v*} \in \mathscr{I}^*(\alpha)$, for each $\alpha \in T_{BA}(Pr(\varphi))$. Now we use these e_i^{v*}'s to define:
  - $\mathscr{R}^v = \{v \xrightarrow{e_i^{v*}} v_i \mid v \xrightarrow{e_i^v} v_i \in \mathscr{R}_w^\varphi\}$.
  - $\mathscr{P}^v = \{(v, e_i^{v*}) \mid \mathscr{P}_w^\varphi(v, e_i^v)\}$.

  Using these sets defined for each state v, we define: $\mathscr{R}^* = \bigcup_{v \in \mathscr{W}^*} \mathscr{R}^v$ and:

  $$\mathscr{P}^* = (\bigcup_{v \in \mathscr{W}^*} \mathscr{P}^v) \cup \{(v, e) \mid v \in \mathscr{W}^* \wedge \mathsf{P}(\alpha) \in L(v) \wedge e \in \mathscr{I}^*(\alpha)\}.$$

- *Define $\mathscr{I}^*(a_i) = \{[\gamma] \mid \vdash_{BA} \gamma \sqsubseteq a_i\}$, for every $a_i \in \Delta_0^*$.*
- *Define $\mathscr{I}^*(p_i) = \mathscr{I}_w^\varphi(p_i)$, for every atomic proposition p_i.*

Let us prove that this new model preserves properties of L.

By the theorem above, we have that $w, M_w^\varphi \vDash \varphi$. As explained above, if we prove that $v, M^* \vDash L(v)$ for every v, we have that $w, M^* \vDash \varphi$. For the states reachable in n steps, we have that $L(v)$ contains only propositional variables or deontic predicaties; for the propositional predicates. the result is trivial. Now, suppose that $\mathsf{P}(\alpha) \in L(v)$, then $v, M_w^\varphi \vDash \mathsf{P}(\alpha)$, and then $v, M^* \vDash \mathsf{P}(\alpha)$, by the definition of M^*. If $\mathsf{P}_\mathsf{w}(\alpha) \in L(v)$, then $v, M_w^\varphi \vDash \mathsf{P}_\mathsf{w}(\alpha)$, and therefore there exists an $e_i \in \mathscr{I}_w^\varphi(\varphi)$ such that $\mathscr{P}_w^\varphi(v, e_i)$, but for this e_i we have a corresponding e_i^v such that $(v, e_i^v) \in \mathscr{P}^*$ and $e_i^v \in \mathscr{I}^*(\alpha)$, and therefore $v, M^* \vDash \mathsf{P}_\mathsf{w}(\alpha)$. If $\neg\mathsf{P}(\alpha) \in L(v)$, then we have an $e_i \in I_w^\varphi$ such that $\neg\mathscr{P}(v, e_i)$; for this e_i, we have an $e_i^v \in \mathscr{P}^*$ and by definition of $\mathscr{P}^*$ we have $\neg\mathscr{P}^*(v, e_i^v)$, since $\neg\mathscr{P}_w^\varphi(v, e_i)$ and $\mathsf{P}(\alpha) \notin L(v)$, otherwise $L(v)$ is inconsistent. Therefore, $v, M^* \vDash \neg\mathsf{P}(\alpha)$. If $\neg\mathsf{P}_\mathsf{w}(\alpha) \in L(v)$, then we have have $\neg\mathscr{P}_w^\varphi(e_i, v)$ for every $e_i \in \mathscr{I}^*(\alpha)$; if we *have $\mathsf{P}(\alpha) \in L(v)$, then $\alpha =_{act} \emptyset$, an equation which is also true in M^* and therefore $v, M^* \vDash \neg \mathsf{P}_{\mathsf{w}}(\alpha)$. If $\alpha \neq_{act} \emptyset$, then $\mathsf{P}(\alpha) \notin L(v)$, and there is no way to introduce a tuple (v, e_i^v) in $\mathscr{P}^*$, so $v, M^* \vDash \neg \mathsf{P}_{\mathsf{w}}(\alpha)$.*

Now, suppose that v is reachable in $k < n$ steps from w; for the deontic predicates and propositional variables the proof proceeds as before. If $\langle \alpha_i \rangle \varphi_i \in L(v)$, then $v, M_w^\varphi \vDash \langle \alpha_i \rangle \varphi_i$, and so there is an $e_i \in \mathscr{I}_w^\varphi(\alpha)$ such that $v \xrightarrow{e_i} v_i$ and $v_i, M_w^\varphi \vDash \varphi_i$. Using induction we get $v_i, M^* \vDash \varphi_i$, and we have, by definition of M^*, that $v \xrightarrow{e_i^v} v_i$; this implies that $v, M^* \vDash \langle \alpha_i \rangle \varphi_i$. If $\neg \langle \beta_i \rangle \psi_i \in L(v)$, then $v, M_w^\varphi \vDash \psi_i$, which means that for all e_i such that $e_i \in \mathscr{I}_w^\varphi(\beta_i)$ and $v \xrightarrow{e_i} v_i$, we have $v_i, M_w^\varphi \vDash \neg \psi$. Since ψ_i is a NF formula, it is a conjunction of formulae, i.e., $\psi_i = \psi_i^1 \wedge \cdots \wedge \psi_i^m$, and, for some of these ψ_i's, we have $v, M_w^\varphi \vDash \neg \psi_i^j$, and by definition of L we have $\neg \psi_i^j \in L(v')$ and therefore, by induction, $v', M^* \vDash \neg \psi_i^j$, which implies $w, M^* \vDash \neg \langle \beta_i \rangle \psi_i$. The theorem follows.

Thus, $\mathsf{D}_\exists(\varphi)$ gives us a bound for the number of new primitive symbols that we need to verify a given formula. From this theorem we get the following corollary:

Corollary 2 *For any DNF formula φ with $\mathsf{D}_\exists(\varphi) = n$, if we have a vocabulary $V = \langle \Delta_0, \Phi_0 \rangle$ and a model M of V such that $w, M \vDash \varphi$, then there exists a model M' of a vocabulary $V' = \langle Pr(\varphi) \cup \{b_1, ..., b_k\}, \Phi_0 \rangle$ such that $w', M' \vDash \varphi$ and $k \leq n$.*
Proof. *Suppose that $w, M \vDash \varphi$ for some M over a vocabulary V. Since φ is in DNF, we know that $\varphi = \varphi_1 \vee ... \vee \varphi_m$ (each φ_i being a NF formulae), and therefore $w, M \vDash \varphi_i$ for some i. By theorem 3 we know that there exists a model M' of a vocabulary $V' = \langle Pr(\varphi) \cup \{b_1, ..., b_k\}, \Phi_0 \rangle$ with $k = \mathsf{D}_\exists(\varphi_i) \leq \mathsf{D}_\exists(\varphi)$ such that $w', M' \vDash \varphi_i$ and then $w', M' \vDash \varphi$. (Note that we can ensure that each $Pr(\varphi) = Pr(\varphi_i)$, by adding the formulae $[a_1 \sqcup ... \sqcup a_t]\top$ to each φ_i with $Pr(\varphi) = \{a_1, ..., a_t\}$, these formulae do not modify the truth value of the former one.)*

Roughly speaking, this theorem says that, if we cannot get a model with n (with $\mathsf{D}_\exists(\varphi) = n$) new primitive actions, we will not get a model by adding further primitive actions to the language. Because each formula is equivalent to a DNF formula, the above result gives us a bound for checking every formula (where the bound depends on the formula under consideration).

The method is as follows: given a formula φ, take its negation and then develop a tableau taking into account at most $\mathsf{D}_\exists(\neg\varphi) = n$ (which can be calculated for any formula, as explained above) primitive actions; if the tableau is closed, then the formula φ is valid for any extension of its vocabulary. It is worth noting that we have two kinds of validities: we have formulae which are valid with respect to one vocabulary (i.e., these formulae are true with respect to all the models of this vocabulary). We can call this notion of validity *local validity*. And we have formulae which are valid with respect to every vocabulary, i.e., a *global validity*. For example the formula $[a \sqcup b]\varphi \leftrightarrow [\mathbf{U}]\varphi$ is valid in the vocabulary $\langle \{a, b\}, \{p, q, s, \dots\} \rangle$ but it is not valid in the vocabulary $\langle \{a, b, c\}, \{p, q, s, \dots\} \rangle$.

5 Time Complexity

We present some results about the time complexity of the method presented above. Our first theorem says that checking local satisfiability is polynomial w.r.t. space.

Theorem 4 *Checking local satisfiability with the tableaux for DPL is PSPACE.*
Proof. *Let us note that, given a vocabulary, the number of its actions is fixed, and then, the number of atomic action terms is constant with respect to formula length. Since the branching is bounded by the number of atoms, it is bounded by a constant, and the length of any branch is linearly bounded by the length of the formula. That is, it is straightforward, given a formula φ, to write a procedure that inspects*

each branch in time $O(a^{|\varphi|})$, where $|\varphi|$ is the length of φ and a is the number of atomic action terms of the given vocabulary. Furthermore, since we only need to inspect a branch at a time, we can develop an algorithm that behaves in a backtracking way; it only needs to keep a record of the current branch, each atom can be described by a binary number of constant length w.r.t. the formula length, and (in each branch) we only use an atom for each existential subformula; i.e., since this branch is linearly bounded by the length of the formula (the number of subformulae of a formula is linear w.r.t. the length of the formula), we only need polynomial space to develop the algorithm.

However, when we want to check if a formula is valid in any possible extension of the vocabulary, we need to use the the results shown in section 4. Interestingly, in this case we also obtain that the method is polynomial w.r.t space.

Theorem 5 *Checking global satisfiability with the tableaux for DPL is PSPACE.*
Proof. *Let us analyze the complexity of the global satisfiability, i.e., when we want to check the satisfiability of a formula in any vocabulary extension. In this case, we add a number of actions to a vocabulary, this number is bounded by the length of the formula. Note that this does not affect the length of branches since we still use one action atom for each existential subformula appearing in the original formula. Moreover, each atom appearing in the branch can be represented using a binary number of a length linearly bounded by the length of the formula (we have a part of the binary number which is constant since it represents the occurrence of the original action, plus a binary number that represents the new actions which are bounded by the length of the formula). And therefore using the same procedure as before, but taking into account the new actions we obtain a PSPACE algorithm.*

Let us compare the complexity of the tableaux method developed here with the complexity of related logics. Boolean modal logic is NExpTime-Complete [17] since the global modality allows us to encode complex problems. Restricting BML to signatures with only a finite number of relation symbols is ExpTime-Complete [17]. That is, the complexity of our tableaux is therefore relatively acceptable w.r.t. the complexity of well-known logics. Taking into account that the modal logic **K** is PSPACE-Complete, we can expect that the complexity of the tableaux is aligned to the complexity of deciding validity in DPL.

Now we give some examples. In figure 3 we build the tableau for the formula: $([\alpha]\varphi \wedge \langle\alpha\rangle\psi) \rightarrow \langle\alpha\rangle(\varphi \wedge \psi)$. This formula is one of the axioms given for dynamic logic in [15]. The crosses at the end of each branch mean that those branches are closed. Note that here we are using a new action symbol.

We have added numbers in the formulae to better explain the example. Formula (1) is the negation of the formula to be proven. Formulae (2) and (3) are obtained by applying the rule *A* to the negation of the implication, formulae (4) and (5) are obtained from formula (2) using the *A* rule. Formulae (6) and (7) follow from formula (3) by application of the *N* rule. In a similar way, we obtained formulae (8) and (9) from formula 4. After that, we have branching using the *P* rule. Finally, we apply the *B* rule to formulae (6) and (7) and we obtain the leaves closing the tableau.

Now, consider the following formula (which is not valid): $\langle\alpha\rangle\varphi \rightarrow [\alpha]\varphi$. The tableau for it is shown in figure 4. Note that, in this case, we use a new action symbol (following corollary 2). First, we reduce the implication, after that we use the rule *P* on the second formula, and then we use rule *P* again in the third formula. We can observe that this tableau has some open branches and using them we can build a “counterexample” (shown in figure 5). Note that we can use the labels in the formulae to put the labels on the transitions, indicating in this way which actions were executed and which were not.

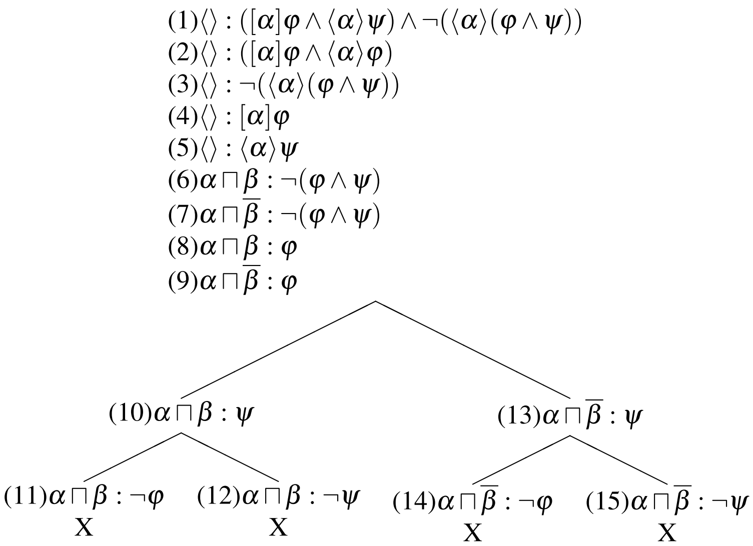
$(1)\langle\rangle : ([\alpha]\varphi \wedge \langle\alpha\rangle\psi) \wedge \neg(\langle\alpha\rangle(\varphi \wedge \psi))$
$(2)\langle\rangle : ([\alpha]\varphi \wedge \langle\alpha\rangle\varphi)$
$(3)\langle\rangle : \neg(\langle\alpha\rangle(\varphi \wedge \psi))$
$(4)\langle\rangle : [\alpha]\varphi$
$(5)\langle\rangle : \langle\alpha\rangle\psi$
$(6)\alpha \sqcap \beta : \neg(\varphi \wedge \psi)$
$(7)\alpha \sqcap \overline{\beta} : \neg(\varphi \wedge \psi)$
$(8)\alpha \sqcap \beta : \varphi$
$(9)\alpha \sqcap \overline{\beta} : \varphi$
$(10)\alpha \sqcap \beta : \psi$
$(13)\alpha \sqcap \overline{\beta} : \psi$
$(11)\alpha \sqcap \beta : \neg\varphi$
X
$(12)\alpha \sqcap \beta : \neg\psi$
X
$(14)\alpha \sqcap \overline{\beta} : \neg\varphi$
X
$(15)\alpha \sqcap \overline{\beta} : \neg\psi$
X

Figure 3: Tableau for $([\alpha]\varphi \wedge \langle\alpha\rangle\psi) \rightarrow \langle\alpha\rangle(\varphi \wedge \psi)$

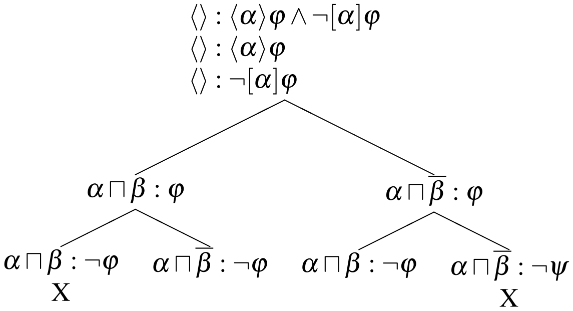
$\langle\rangle : \langle\alpha\rangle\varphi \wedge \neg[\alpha]\varphi$
$\langle\rangle : \langle\alpha\rangle\varphi$
$\langle\rangle : \neg[\alpha]\varphi$
$\alpha \sqcap \beta : \varphi$
$\alpha \sqcap \overline{\beta} : \varphi$
$\alpha \sqcap \beta : \neg\varphi$
X
$\alpha \sqcap \overline{\beta} : \neg\varphi$
$\alpha \sqcap \beta : \neg\varphi$
$\alpha \sqcap \overline{\beta} : \neg\psi$
X

Figure 4: Tableau for $\langle\alpha\rangle\varphi \rightarrow [\alpha]\varphi$

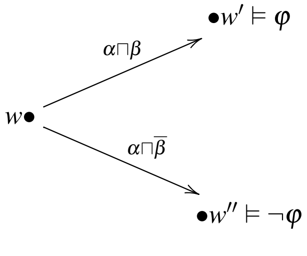
$\bullet w' \vDash \varphi$
$\alpha \sqcap \beta$
$w\bullet$
$\alpha \sqcap \overline{\beta}$
$\bullet w'' \vDash \neg\varphi$

Figure 5: Counterexample for $\langle\alpha\rangle\varphi \rightarrow [\alpha]\varphi$

6 Further Remarks

In this paper we have investigated further the properties of the tableaux system presented in [4]. One of the main features of the system is that it uses the underlying algebra of actions to produce tableaux, enabling it to manage successfully the intersection and complement operators on actions. Moreover, the algebra of actions allows us to extend the propositional tableaux system to manage temporal predicates.

A relevant point demonstrated in the paper is that, though we have a finite number of actions, it is possible to prove properties which are valid in any extension of the actual vocabulary, which seems to be very useful in practice to verify computing components which could be part of bigger systems. This kind of completeness with respect to language extension is also preserved for the temporal version of the logic. In addition, we have investigated the time complexity of the tableaux system and we prove that it is in PSPACE in comparison with related logics, which are in EXPTIME.

This deontic logic was presented in [5], where several of its properties, and some examples of application, were shown. In those papers we proposed this logic to specify and verify properties related to fault-tolerance. It seems very useful to apply the tableaux system described here to such examples. We leave this as further work.

References

[1] P. Blackburn, M.de Rijke & Y.de Venema (2001): *Modal Logic*. Cambridge Tracts in Theoretical Computer Science 53.

[2] J. Broersen (2003): *Modal Action Logics for Reasoning about Reactive Systems*. Ph.D. thesis, Vrije University.

[3] Pablo F. Castro (2009): *Deontic Action Logics for the Specification and Analysis of Fault-Tolerance*. Ph.D. thesis, McMaster University, Department of Computing and Software.

[4] Pablo F. Castro & T.S.E. Maibaum (2008): *A Tableaux System for Deontic Action Logic*. In: Proceedings of 9th International Conference on Deontic Logic in Computer Science , Luxembourg., Springer-Verlag, doi:10.1007/978-3-540-70525-3_4.

[5] Pablo F. Castro & T.S.E. Maibaum (2009): *Deontic Action Logic, Atomic Boolean Algebra and Fault-Tolerance*. Journal of Applied Logic 7(4), pp. 441–466, doi:10.1016/j.jal.2009.02.001.

[6] Brian F. Chellas (1999): *Modal Logic: An Introduction*. Cambridge University Press.

[7] E.A. Emerson (1995): *Temporal and Modal Logic*. In: Handbook of Theorical Computer Science, Elsevier, pp. 995–1072.

[8] Kit Fine (1975): *Normal Forms in Modal Logics*. In: Notre Dame Journal of Formal Logic, XVI, pp. 229–237, doi:10.1305/ndjfl/1093891703.

[9] M. Fitting (1990): *First-Order Logic and Automated Theorem Proving*. Springer-Verlag, doi:10.1007/978-1-4684-0357-2.

[10] M. Fitting (April 1972): *Tableau Methods of Proof for Modal Logics*. In: Notre Dame Journal of Formal Logic, XIII, doi:10.1305/ndjfl/1093894722.

[11] Dov M. Gabbay (1996): *Labelled Deductive Systems, Volume 1*. Oxford University Press.

[12] G. Gargov & S. Passy (1990): *A Note on Boolean Logic*. In P.P.Petkov, editor: Proceedings of the Heyting Summerschool, Plenum Press, doi:10.1007/978-1-4613-0609-2_21.

[13] G. Giacomo & F. Massacci (1996): *Tableaux and Algorithms for Propositional Dynamic Logic with Converse*. In: Conference on Automated Deduction, doi:10.1006/inco.1999.2852.

[14] Rajeev Goré (1995): *Tableau Methods for Modal and Temporal Logics*. Technical Report TR-ARP-15-95, Australian National University.

[15] D. Harel, D. Kozen & J. Tiuryn (2000): *Dynamic Logic*. MIT Press.

[16] S. Khosla & T.S.E. Maibaum (1985): *The Prescription and Description of State-Based Systems.* In H.Barringer B.Banieqnal & A.Pnueli, editors: *Temporal Logic in Computation*, Springer-Verlag, doi:10.1007/3-540-51803-7_30.

[17] Carsten Lutz & Ulrike Sattler (2000): *The Complexity of Reasoning with Boolean Modal Logics*. In: *Advances in Modal Logic 3*, World Scientific, pp. 329–348.

[18] Fabio Massacci (2000): *Single Step Tableaux for Modal Logics*. *J. Autom. Reasoning* 24(3), pp. 319–364, doi:10.1023/A:1006155811656.

[19] J.J. Meyer (1988): *A Different Approach to Deontic Logic: Deontic Logic Viewed as Variant of Dynamic Logic*. In: *Notre Dame Journal of Formal Logic*, 29, doi:10.1305/ndjfl/1093637776.

[20] J.D. Monk (1976): *Mathematical Logic*. Graduate Texts in Mathematics, Springer-Verlag.

[21] V.R. Pratt (1978): *A Practical Decision Method for Propositional Dynamic Logic*. ACM Symposium on Theory of Computing, doi:10.1145/800133.804362.

[22] Krister Segerberg (1982): *A Deontic Logic of Action*. *Studia Logica* 41, pp. 269–282, doi:10.1007/BF00370348.

[23] R.M. Smullyan (1968): *First-Order Logic*. Springer-Verlag New York, doi:10.1007/978-3-642-86718-7.

[24] Robert Trypuz & Piotr Kulicki (2010): *Towards Metalogical Systematisation of Deontic Action Logics Based on Boolean Algebra*. In: *Deontic Logic in Computer Science, 10th International Conference*, Lecture Notes in Computer Science 6181 Springer, doi:10.1007/978-3-642-14183-6_11.