On the Complexity of Dynamic Epistemic Logic

Guillaume Aucher
University of Rennes 1 - INRIA
guillaume.aucher@irisa.fr

François Schwarzentruber
ENS Cachan - Brittany extension
francois.schwarzentruber@bretagne.ens-cachan.fr

ABSTRACT

Although Dynamic Epistemic Logic (DEL) is an influencial logical framework for representing and reasoning about information change, little is known about the computational complexity of its associated decision problems. As a matter of fact, we only know that for public announcement logic, a fragment of DEL, the satisfiability problem and the model-checking problem are respectively PSPACE-complete and in P. We contribute to fill this gap by proving that for the DEL language with event models, the model-checking problem is, surprisingly, PSPACE-complete. Also, we prove that the satisfiability problem is NEXPTIME-complete. In doing so, we provide a sound and complete tableau method deciding the satisfiability problem.

Categories and Subject Descriptors

I.2.4 [**Knowledge representation formalisms and methods**]: Modal logic; F.1.3 [**Complexity measure and classes**]: Reducibility and completeness

General Terms

Theory

Keywords

Dynamic epistemic logic, computational complexity, model checking, satisfiability

1. INTRODUCTION

Research fields like distributed artificial intelligence, distributed computing and game theory all deal with groups of human or non-human agents which interact, exchange and receive information. The problems they address range from multi-agent planning and design of distributed protocols to strategic decision making in groups. In order to address appropriately and rigorously these problems, it is necessary to be able to provide formal means for representing and reasoning about such interactions and flows of information. The framework of Dynamic Epistemic Logic (DEL for short) is very well suited to this aim. Indeed, it is a logical framework where one can represent and reason about beliefs and knowledge change of multiple agents, and more generally about information change.

The theoretical work of the above mentionned research fields has already been applied to various practical problems stemming from telecommunication networks, world wide web, peer to peer networks, aircraft control systems, and so on... In general, in all applied contexts, the investigation of the algorithmic aspects of the formalisms employed plays an important role in determining whether and to which extent they can be applied. For this reason, the algorithmic aspects of DEL need to be studied.

To this aim, a preliminary step consists in studying the computational properties of its main associated decision problems, namely the model checking problem and the satisfiability problem. Indeed, it will indirectly provide algorithmic methods to solve these decision problems and give us a hint of whether and to which extent our methods can be applied. However, surprisingly, little is known about the computational complexity of these problems. We only know that for public announcement logic, a fragment of DEL [Plaza, 1989], the model checking problem is in P and the satisfiability problem is PSPACE-complete. Here, we aim to fill this gap for the full language of DEL with event models.

DEL is built on top of epistemic logic. An epistemic model represents how a given set of agents perceive the actual world in terms of beliefs and knolwedge about this world and about the other agents' beliefs. The insight of the DEL approach is that one can describe how an event is perceived by agents in a very similar way: an agent's perception of an event can also be described in terms of beliefs and knowledge. For example, at the battle of Waterloo, when marshal Blücher reveived the message of the duke of Wellington inviting him to join the attack at dawn against Napoleon, Wellington did not *know* at that very moment that Blücher was receiving his message, and Blücher *knew* it. This is a typical example of announcement which is not public. This led Baltag, Moss and Solecki to introduce the notion of *event model* [Baltag et al., 1998]. The definition of an event model, denoted $(\mathcal{M}', w')$, is very similar to the definition of an epistemic model. They also introduced a *product update*, which defines a new epistemic model representing the situation after the event. Then, they extended the epistemic language with dynamic operators $[\mathcal{M}', w']\varphi$ standing for 'φ holds after the occurence of the event represented by $(\mathcal{M}', w')$'.

Using the so-called reduction axioms, it turns out that any formula with dynamic operator(s) can be translated to an equivalent epistemic formula without dynamic operator. As a first approximation, we could be tempted to use these reduction axioms to reduce both the model checking problem and the satisfiability problem of DEL to the model checking problem and the satisfiability problem of epistemic logic, because optimal algorithmic methods already exist for these related problems. However, the reduction algorithm induced by the reduction axioms is exponential in the size of the input formula. Therefore, for the satisfiability problem, we only obtain an algorithm which is in EXPSPACE (because the satisfiability problem of epistemic logic is PSPACE-complete), and for the model checking problem, we only obtain an algorithm which is in EXPTIME (because the model checking problem of epistemic logic is in P). These algorithms are not optimal because, as we shall see, there exist an algorithm solving the satisfiability problem which is in NEXPTIME⊆ EXPSPACE and also an algorithm solving the model checking problem which is in PSPACE⊆ EXPTIME. Our algorithm for solving the satisfiability problem is based on a sound and complete tableau method which does not resort to the reduction axioms.

The paper is organized as follows. In Section 2, we recall the core of the DEL framework and the different variants of languages with event models which have been introduced in the literature. In Section 3, we prove that the model checking problem of DEL is PSPACE-complete, and in Section 4 we prove that the satisfiability problem is NEXPTIME-complete. In Section 5, we discuss related works and whether our results still hold when we extend the expressivity of the language with common belief and 'star' iteration operators. We conclude in Section 6.

2. DYNAMIC EPISTEMIC LOGIC

Following the methodology of DEL, we split the exposition of the DEL logical framework into three subsections. In Section 2.1, we recall the syntax and semantics of the epistemic language. In Section 2.2, we define event models, and in Section 2.3, we define the product update. In Section 2.4, we recall the different languages that have been introduced in the DEL literature and we introduce our language $\mathcal{L}_{DEL}$.

2.1 Epistemic language

In the rest of the paper, ATM is a countable set of atomic propositions and AGT is a finite set of agents.

A (pointed) epistemic model $(\mathcal{M}, w)$ represents how the actual world represented by w is perceived by the agents. Intuitively, in this definition, $vR_a u$ means that in world v agent a considers that world u might be the actual world.

Definition 1 (Epistemic model).
An *epistemic model* is a tuple $\mathcal{M} = (W, R, V)$ where W is a non-empty set of possible worlds, R maps each agent $a \in AGT$ to a relation $R_a \subseteq W \times W$ and $V : ATM \to 2^W$ is a function called a valuation. We abusively write $w \in \mathcal{M}$ for $w \in W$ and we say that $(\mathcal{M}, w)$ is a *pointed epistemic model*. We also write $v \in R_a(w)$ for $wR_a v$.

Then, we define the following epistemic language $\mathcal{L}_{EL}$. It can be used to describe and state properties of epistemic models:

Definition 2 (Epistemic language).

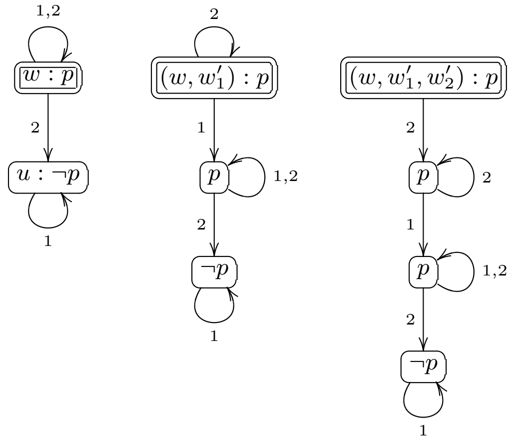

Figure 1: Pointed epistemic models $(\mathcal{M}, w)$ (*left*), $((\mathcal{M} \otimes \mathcal{M}'_1), (w, w'_1))$ (*center*) and $(\mathcal{M} \otimes \mathcal{M}'_1 \otimes \mathcal{M}'_2, (w, w'_1, w'_2))$ (*right*)

The *language* $\mathcal{L}_{EL}$ of epistemic logic is defined as follows:

$$\mathcal{L}_{EL} : \varphi \ ::= \ p \ | \ \neg\varphi \ | \ (\varphi \land \varphi) \ | \ B_a\varphi$$

where $p \in ATM$ and $a \in AGT$. A formula of $\mathcal{L}_{EL}$ is called an *epistemic formula*. The formula $\bot$ is an abbreviation for $p \land \neg p$, and $\top$ is an abbreviation for $\neg\bot$. The formula $\langle B_a \rangle\varphi$ is an abbreviation of $\neg B_a \neg\varphi$. The *size of a formula* $\varphi \in \mathcal{L}_{EL}$ is defined by induction as follows: $|p| = 1$; $|\neg\varphi| = 1 + |\varphi|$; $|\varphi \land \psi| = 1 + |\varphi| + |\psi|$; $|B_a\varphi| = 1 + |\varphi|$.

Intuitively, the formula $B_a\varphi$ reads as 'agent a believes that φ holds in the current situation'.

Definition 3 (Truth conditions).
Given an epistemic model $\mathcal{M} = (W, R, V)$ and a formula $\varphi \in \mathcal{L}_{EL}$, we define inductively the satisfaction relation $\models \subseteq W \times \mathcal{L}_{EL}$ as follows: for all $w \in W$,

$\mathcal{M}, w \models p$	iff	$w \in V(p)$
$\mathcal{M}, w \models \varphi \land \psi$	iff	$\mathcal{M}, w \models \varphi$ and $\mathcal{M}, w \models \psi$
$\mathcal{M}, w \models \neg\varphi$	iff	not $\mathcal{M}, w \models \varphi$
$\mathcal{M}, w \models B_a\varphi$	iff	for all $v \in R_a(w)$, we have $\mathcal{M}, v \models \varphi$

We write $\mathcal{M} \models \varphi$ when for all $w \in \mathcal{M}$, it holds that $\mathcal{M}, w \models \varphi$. Also, we write $\models \varphi$, and we say that φ is *valid*, when for all epistemic model $\mathcal{M}$, it holds that $\mathcal{M} \models \varphi$. Dually, we say that φ is *satisfiable* when $\neg\varphi$ is not valid.

Example 1. *Our running example is inspired by the coordinated attack problem from the distributed systems folklore [Fagin et al., 1995]. Our set of atomic propositions is $ATM = \{p\}$ and our set of agents is $AGT = \{1, 2\}$. Agent 1 is the duke of Wellington and agent 2 is marshal Blücher; p stands for 'Wellington wants to attack at dawn'. The initial situation is represented in Figure 1 by the pointed epistemic model $(\mathcal{M}, w) = (\{w, u\}, R_1 = \{(w, w), (u, u)\}, R_2 = \{(w, w), (w, u)\}, V(p) = \{w\})$. In this pointed epistemic model, it holds that $\mathcal{M}, w \models p \land B_1 p$: Wellington 'knows' that he wants to attack at dawn. It also holds that $\mathcal{M}, w \models \neg B_2 p$: Blücher does not 'know' that Wellington wants to attack at dawn; and $\mathcal{M}, w \models B_1 \neg B_2 p$: Wellington 'knows' that Blücher does not 'know' that he wants to attack at dawn.*

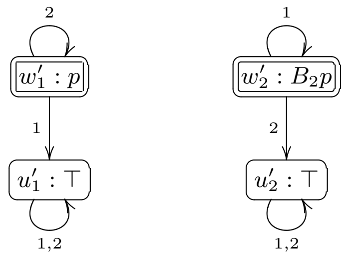

Figure 2: Pointed event models $(\mathcal{M}'_1, w'_1)$ (*left*) and $(\mathcal{M}'_2, w'_2)$ (*right*)

2.2 Event model

A (pointed) event model $(\mathcal{M}', w')$ represents how the actual event represented by w' is perceived by the agents. Intuitively, in this definition, $u' R'_a v'$ means that while the possible event represented by u' is occurring, agent a considers possible that the possible event represented by v' is actually occurring.

Definition 4 (Event model).
An *event model* is a tuple $\mathcal{M}' = (W', R', Pre)$ where W' is a non-empty and finite set of possible events, R' maps each agent $a \in AGT$ to a relation $R'_a \subseteq W' \times W'$ and $Pre : W' \to \mathcal{L}_{EL}$ is a function that maps each event to a precondition expressed in the epistemic language.

We abusively write $w' \in \mathcal{M}'$ for $w' \in W'$ and we say that $(\mathcal{M}', w')$ is a *pointed event model*. The *size of an event model* $\mathcal{M}' = (W', R', Pre)$ is noted $|\mathcal{M}'|$ and is defined as follows: $card(W') + \sum_{a \in AGT} card(R'_a) + \sum_{w' \in W'} |Pre(w')|$.

Example 2. *In Figure 2 are represented two pointed event models. The first, $(\mathcal{M}_1, w'_1) = (\{w'_1, u'_1\}, R_1 = \{(w'_1, u'_1), (u'_1, u'_1)\}, R_2 = \{(w'_1, w'_1), (u'_1, u'_1)\}, Pre, w'_1)$ where $Pre(w'_1) = p$ and $Pre(u'_1) = \top$, represents the event whereby Blücher receives the message of Wellington that he wants to attack at dawn. When this happens, Wellington believes that nothing happens and believes that this is even common knowledge. The second, $(\mathcal{M}_2, w'_2) = (\{w'_2, u'_2\}, R_1 = \{(w'_2, w'_2), (u'_2, u'_2)\}, R_2 = \{(w'_2, u'_2), (u'_2, u'_2)\}, Pre, w'_2)$, where $Pre(w'_2) = B_2 p$ and $Pre(u'_2) = \top$, represents the event whereby Wellington receives the message of Blücher telling him that he 'knows' that Wellington wants to attack at dawn.*

2.3 Product update

The following product update yields a new pointed epistemic model $\mathcal{M} \otimes \mathcal{M}', (w, w')$ representing how the new situation which was previously represented by $(\mathcal{M}, w)$ is perceived by the agents after the occurrence of the event represented by $(\mathcal{M}', w')$.

Definition 5 (Product update).
Let $\mathcal{M} = (W, R, V)$ be an epistemic model and let $\mathcal{M}' = (W', R', Pre)$ be an event model. The *product update of $\mathcal{M}$ by $\mathcal{M}'$* is the epistemic model $\mathcal{M} \otimes \mathcal{M}' = (W'', R'', V'')$ defined as follows (p and a range over ATM and AGT respectively):

$$W'' = \{(w, w') \in W \times W' \mid \mathcal{M}, w \models Pre(u')\}$$
$$R''_a = \{\langle (w, w'), (v, v') \rangle \in W'' \times W'' \mid wR_a v \text{ and } w' R'_a v'\}$$
$$V''(p) = \{(w, w') \in W'' \mid w \in V(p)\}$$

Given a pointed epistemic model $(\mathcal{M}, w)$, and a pointed event model $(\mathcal{M}', w')$, we say that $(\mathcal{M}', w')$ is *executable* in $(\mathcal{M}, w)$ when $\mathcal{M}, w \models Pre(w')$. If $\mathcal{M}$ is an epistemic model and $\mathcal{M}'_1, \ldots, \mathcal{M}'_n$ are event models, we abusively write $\mathcal{M} \otimes \mathcal{M}'_1 \otimes \cdots \otimes \mathcal{M}'_n$ for $(\ldots((\mathcal{M} \otimes \mathcal{M}'_1) \otimes \mathcal{M}'_2) \otimes \ldots) \otimes \mathcal{M}'_n$ and $(w, w'_1, \ldots, w'_n)$ for $(\ldots((w, w'_1), w'_2), \ldots), w'_n)$.

Example 3. *The pointed epistemic models $((\mathcal{M} \otimes \mathcal{M}'_1), (w, w'_1))$ and $(\mathcal{M} \otimes \mathcal{M}'_1 \otimes \mathcal{M}'_2, (w, w'_1, w'_2))$ are represented in Figure 1. After Blücher receives the message of Wellington, Blücher 'knows' that Wellington wants to attack at dawn, but Wellington does not 'know' that Blücher 'knows' it: $\mathcal{M} \otimes \mathcal{M}'_1, (w, w'_1) \models p \wedge B_2 p \wedge \neg B_1 B_2 p$. Likewise, after Wellington receives the message of Blücher telling him that he 'knows' that he wants to attack at dawn ($B_2 p$), Wellington 'knows' that Blücher 'knows' that he wants to attack at dawn, but Blücher does not 'know' that Wellington 'knows' it: $\mathcal{M} \otimes \mathcal{M}'_1 \otimes \mathcal{M}'_2, (w, w'_1, w'_2) \models p \wedge B_2 p \wedge B_1 B_2 p \wedge \neg B_2 B_1 B_2 p$. Hence, in particular, $\mathcal{M}, w \models \neg[\mathcal{M}'_1, w'_1][\mathcal{M}'_2, w'_2] B_2 B_1 B_2 p$.*

2.4 Languages of DEL

In [Baltag et al., 1998], the language is defined as follows:

$$\varphi \ ::= \ p \mid \neg\varphi \mid (\varphi \wedge \varphi) \mid B_a \varphi \mid [\mathcal{M}', w']\varphi$$

where $p \in ATM$, $a \in AGT$ and $(\mathcal{M}', w')$ is any pointed and finite event model. The formula $\langle \mathcal{M}', w' \rangle \varphi$ is an abbreviation for $\neg[\mathcal{M}', w']\neg\varphi$.

Intuitively, $[\mathcal{M}', w']\varphi$ reads as 'φ will hold after the occurence of the event represented by $(\mathcal{M}', w')$' and $\langle \mathcal{M}', w' \rangle \varphi$ reads as 'the event represented by $(\mathcal{M}', w')$ is executable in the current situation and φ will hold after its execution'.

However, note that in this definition, preconditions of event models are necessarily epistemic formulas. In [Baltag and Moss, 2004], another language is introduced which can deal with event models whose preconditions may involve formulas with event models. This language relies on the notion of *event signature* and the epistemic language is extended with a modality $[\Sigma, \varphi_1, \ldots, \varphi_n]\varphi$, where Σ is an event signature. The language of [Baltag and Moss, 2004] also includes PDL-like program constructions such as sequential composition, union and 'star' operation of event models (see Section 5 for a definition of these program constructions).

In [van Ditmarsch et al., 2007], preconditions can also be formulas involving event models, but only union of programs is allowed. It is therefore a fragment of the language of [Baltag and Moss, 2004] since it does not include sequential composition nor the 'star' operation. This will be our language in this paper.

Definition 6 ([van Ditmarsch et al., 2007]).
The language $\mathcal{L}_{DEL}$ is the union of the *formulas* $\varphi \in \mathcal{L}^{stat}_{\otimes}$ and the *events* (or *epistemic events*) $\pi \in \mathcal{L}^{dyn}_{\otimes}$ defined by the following rule:

$$\mathcal{L}^{stat}_{\otimes} : \varphi \ ::= p \mid \neg\varphi \mid (\varphi \wedge \varphi) \mid B_a \varphi \mid [\pi]\varphi$$
$$\mathcal{L}^{dyn}_{\otimes} : \pi \ ::= \mathcal{M}', w' \mid (\pi \cup \pi)$$

where $p \in ATM$, $a \in AGT$ and $(\mathcal{M}', w')$ is a pointed and finite event model such that for all $w' \in \mathcal{M}'$, $Pre(w')$ is a formula of $\mathcal{L}^{stat}_{\otimes}$ that has already been constructed in a previous stage of the inductively defined hierarchy.

The *size of* $\varphi \in \mathcal{L}_{DEL}$ is defined as for the epistemic language together with the induction case $|[\pi]\varphi| = 1 + |\pi| + |\varphi|$ where $|\mathcal{M}', w'| = |\mathcal{M}'|$, and $|\pi \cup \gamma| = 1 + |\pi| + |\gamma|$.

Definition 7 (Truth conditions).
Given an epistemic model $\mathcal{M} = (W, R, V)$ and a formula $\varphi \in \mathcal{L}_{DEL}$, we define inductively the satisfaction relation $\models \subseteq W \times \mathcal{L}_{DEL}$ as follows:

$$\begin{array}{lll} \mathcal{M}, w \models [\mathcal{M}', w']\varphi & \text{iff} & \mathcal{M}, w \models Pre(w') \text{ implies} \\ & & \mathcal{M} \otimes \mathcal{M}', (w, w') \models \varphi \\ \mathcal{M}, w \models [\pi \cup \gamma] & \text{iff} & \mathcal{M}, w \models [\pi]\varphi \text{ and } \mathcal{M}, w \models [\gamma]\varphi. \end{array}$$

The other induction steps are identical to the induction steps of Definition 3.

The results in this paper are the same whether or not the formulas of the preconditions involve event models. However, the result of NEXPTIME-completeness of the satisfiability problem of Section 4 holds only if we consider union of event models as a program construction in the language.

3. MODEL CHECKING PROBLEM

The *model checking problem of* $\mathcal{L}_{DEL}$ is defined as follows:

> **Input**: a pointed epistemic model $(\mathcal{M}, w)$ and a formula $\varphi \in \mathcal{L}_{DEL}$;
>
> **Output**: yes iff $\mathcal{M}, w \models \varphi$.

Whereas the model checking problem with an epistemic formula of $\mathcal{L}_{EL}$ is in P, model checking with a formula of $\mathcal{L}_{DEL}$ is surprisingly PSPACE-complete. This shows that the addition of dynamic modalities with event models to $\mathcal{L}_{EL}$ increases tremendously the expressivity of the language.

3.1 Upper bound

In Figure 3 is defined a deterministic algorithm M-Check($\mathcal{M}$, $\mathcal{M}'_1, \dots, \mathcal{M}'_i, w, u_1, \dots, u_i, \varphi$) that checks whether we have $\mathcal{M} \otimes \mathcal{M}'_1 \otimes \dots \mathcal{M}'_i, (w, u'_1, \dots, u'_i) \models \varphi$, where $(\mathcal{M}, w)$ is a pointed epistemic model and for all $j \in \{1, \dots, i\}$, $(\mathcal{M}'_j, u'_j)$ is a pointed event model. The precondition of a call to the function M-Check($\dots$) is that $(w, u'_1, \dots, u'_i) \in \mathcal{M} \otimes \mathcal{M}'_1 \otimes \dots \mathcal{M}'_i$, that is, the sequence $(\mathcal{M}'_1, u'_1) \dots, (\mathcal{M}'_i, u'_i)$ is executable in $(\mathcal{M}, w)$. In order to check whether $\mathcal{M}, w \models \varphi$, we just call M-Check($\mathcal{M}, w, \varphi$).

Theorem 1. *The model checking problem of $\mathcal{L}_{DEL}$ is in PSPACE.*

Proof sketch. Terminaison and correction of the algorithm M-Check are easily proved over the size of the input defined by $|\mathcal{M}| + \sum_{k=1}^{i} |\mathcal{M}'_k| + |\varphi|$. As for complexity, the algorithm requires a polynomial amount of space in the size of the input. Indeed, as the size of the input is strictly decreasing at each recursive call, the number of recursive calls in the call stack is linear in the size of the input. Then, each of the current call requires a polynomial amount of space in the size of the input for storing the value of local variables: the most consuming case is $B_a\psi$ where we have to save all the current values of u, $u_1, \dots, u_i$ in the loop **for**. □

3.2 Lower bound

We prove that the algorithm of the previous section is optimal. To do so, we provide a polynomial reduction of the *quantified Boolean formula satisfiability problem*, known to be PSPACE-complete [Papadimitriou, 1995, p. 455] to the model-checking problem of $\mathcal{L}_{DEL}$.

```
function M-Check(w  M'_1, w'_1; ...; M'_i, w'_i  φ)
  match (φ)
    case p:
      return w ∈ V(p);
    case ¬ψ:
      return not M-Check(w  M'_1, w'_1; ...; M'_i, w'_i  ψ);
    case ψ_1 ∧ ψ_2:
      return (M-Check(w  M'_1, w'_1; ...; M'_i, w'_i  ψ_1) and
        M-Check(w  M'_1, w'_1; ...; M'_i, w'_i  ψ_2));
    case B_a ψ:
      for u ∈ R_a(w)
      for u'_1 ∈ R'_a(w'_1)
      if M-Check(w, Pre(u'_1))
      :
      :
      for u'_i ∈ R'_a(w'_i)
      if M-Check(u  M'_1, u'_1; ...; M'_{i-1}, u'_{i-1}  Pre(u'_i))
        if not M-Check(u  M'_1, u'_1; ...; M'_i, u'_i  ψ);
          return false ;
        endIf
      endIf endFor ... endIf endFor endFor
      return true ;
    case [M', w']ψ:
      if M-Check(w  M'_1, w'_1; ...; M'_i, w'_i  Pre(w'))
        return M-Check(w  M'_1, w'_1; ...; M'_i, w'_i; M', w'  ψ);
      endIf
      return true ;
    case [π ∪ γ]ψ:
      return (M-Check(w  M'_1, w'_1; ...; M'_i, w'_i  [π]ψ) and
        M-Check(w  M'_1, w'_1; ...; M'_i, w'_i  [γ]ψ));
  endMatch
endFunction
```

Figure 3: PSPACE algorithm for model checking

Theorem 2. *The model checking problem of $\mathcal{L}_{DEL}$ is PSPACE-hard.*

Proof. Without loss of generality, we only consider in this proof quantified Boolean formulas of the form $\forall p_1 \exists p_2 \forall p_3 \dots \forall p_{2k-1} \exists p_{2k} \psi(p_1, \dots p_{2k})$, where $\psi(p_1, \dots, p_{2k})$ is a Boolean formula over the atomic propositions $p_1, \dots, p_{2k}$. The formula $\forall p_1 \exists p_2 \forall p_3 \dots \forall p_{2k-1} \exists p_{2k} \psi(p_1, \dots p_{2k})$ is *satisfiable* iff for both truth values of the atomic proposition p_1 there is a truth value for the atomic proposition p_2 such that for both truth values of the atomic proposition p_3, and so on up to p_{2k}, the formula $\psi(p_1, \dots p_{2k})$ is true in the overall truth assignment.

We can restrict ourselves to $\mathcal{L}_{DEL}$ where there is only one agent a. The *quantified Boolean formula satisfiability problem* is defined as follows:

> **Input**: a natural number k and a quantified Boolean formula $\varphi \triangleq \forall p_1 \exists p_2 \forall p_3 \dots \forall p_{2k-1} \exists p_{2k} \psi(p_1, \dots, p_{2k})$;
>
> **Output**: yes iff φ is satisfiable.

Let $\varphi = \forall p_1 \exists p_2 \forall p_3 \dots \forall p_{2k-1} \exists p_{2k} \psi(p_1, \dots p_{2k})$ be a quantified Boolean formula. We define a pointed epistemic model $\mathcal{M}, w^0$, $2k$ pointed event models $\mathcal{M}'_1, w'^0_1, \dots, \mathcal{M}'_{2k}, w'^0_{2k}$, a pointed event model $\mathcal{M}'_\circlearrowleft, w'^0_\circlearrowleft$ and an epistemic formula ψ' that are computable in polynomial time in the size of φ such that:

$$\varphi \text{ is satisfiable in quantified Boolean logic}$$
$$\text{iff}$$
$$\mathcal{M}, w^0 \models [\mathcal{M}'_1, w'^0_1 \cup \mathcal{M}'_\circlearrowleft, w'^0_\circlearrowleft]\langle \mathcal{M}'_2, w'^0_2 \cup \mathcal{M}'_\circlearrowleft, w'^0_\circlearrowleft \rangle \dots$$
$$[\mathcal{M}'_{2k-1}, w'^0_{2k-1} \cup \mathcal{M}'_\circlearrowleft, w'^0_\circlearrowleft]\langle \mathcal{M}'_{2k}, w'^0_{2k} \cup \mathcal{M}'_\circlearrowleft, w'^0_\circlearrowleft \rangle \psi'.$$

The corresponding instance of the model checking problem of $\mathcal{L}_{DEL}$ is computable in polynomial time in the size of φ. Now, let us describe $\mathcal{M}, w_0$, the event models $\mathcal{M}'_1, w_1'^0, \dots, \mathcal{M}'_{2k}, w_{2k}'^0$, the event model $\mathcal{M}'_{\circlearrowright}, w_{\circlearrowright}'^0$ and ψ'.

- $\mathcal{M} = (W, R, V)$ is defined by:
  - $W = \{w^0, w^1, \dots, w^{2k+1}\}$;
  - $R_a = \{(w^j, w^{j+1} \mid j \in \{0, \dots, 2k\}\}$;
  - and $V(p) = \emptyset$ for all $p \in ATM$
- For all $i \in \{1, \dots, 2k\}$, $\mathcal{M}'_i = (W'_i, R'_i, Pre_i)$ is defined by:
  - $W'_i = \{w_i'^0, w_i'^1, \dots, w_i'^i, w_i'^{\circlearrowright}\}$
  - ${R_i}'_a = \{(w_i'^j, w_i'^{j+1} \mid j \in \{0, \dots, i\}\} \cup \{(w_i'^0, w_i'^{\circlearrowright}), (w_i'^{\circlearrowright}, w_i'^{\circlearrowright})\}$
  - and $Pre_i(u') = \top$ for all $u' \in W'_i$
- $\mathcal{M}'_{\circlearrowright}, w_{\circlearrowright}'^0 = (W'_{\circlearrowright}, R'_{\circlearrowright}, Pre_{\circlearrowright})$ is defined by:
  - $W'_{\circlearrowright} = \{w_{\circlearrowright}'^0\}$
  - ${R_{\circlearrowright}}'_a = \{(w_{\circlearrowright}'^0, w_{\circlearrowright}'^0)\}$
  - $Pre_{\circlearrowright}(w_{\circlearrowright}'^0) = \top$
- $\psi' = \psi(p_1 \leftarrow \langle B_a \rangle B_a \bot, \dots, p_{2k} \leftarrow (\langle B_a \rangle)^{2k} B_a \bot)$, that is, ψ' is the formula ψ where all p_i occurrences are substituted by $(\langle B_a \rangle)^i B_a \bot$.

The semantics is simulated in the following way. The proposition p_i is interpreted as the presence of a chain of length exactly i from the root of a given epistemic model. That is why in ψ', the proposition p_i is substituted by $(\langle B_a \rangle)^i B_\bot$, which is true in the root of the final epistemic model iff there exists a chain of length i in that model.

Note that updating an epistemic model where there is a chain of length $2k+1$ by $\mathcal{M}'_i, w_i'^0$ where $i \in \{1, \dots, 2k\}$:

- preserves the presence or absence of any chain of length $j \neq i$; in particular, it always preserves the presence of the chain of length $2k+1$;
- adds a chain of length i, that is p_i becomes true;

Note also that updating an epistemic model where there is a chain of length $2k+1$ by $\mathcal{M}'_{\circlearrowright}, w_{\circlearrowright}'^0$ preserves the presence or absence of any chain. So, it will keep p_i false if it was already false and it will keep any p_i true if it was already true. In other words, the $\mathcal{M}'_{\circlearrowright}, w_{\circlearrowright}'^0$ is a neutral element for the product update.

The crucial invariant property (Inv) of an epistemic model is the existence of a chain of length $2k+1$ in any update of $\mathcal{M}, w^0$ by any sequence of $\mathcal{M}'_{\circlearrowright}, w_{\circlearrowright}'^0$ and $\mathcal{M}'_i, w_i'^0$.

The behavior of $\forall p_i$ in quantified Boolean logic consists in a universal choice of a truth value for p_i. It is translated by the update operator $[\mathcal{M}'_i, w_i'^0 \cup \mathcal{M}'_{\circlearrowright}, w_{\circlearrowright}'^0]$ whose semantics is to choose *universally* the update of the epistemic model by $\mathcal{M}'_i, w_i'^0$, that will give a new updated epistemic model with a chain of length i, that is p_i is true, or by $\mathcal{M}'_{\circlearrowright}, w_{\circlearrowright}'^0$ that will let the new updated epistemic model without a chain of length i, that is p_i is false.

The behavior of $\exists p_i$ in quantified Boolean logic consists in an existential choice of a truth value for p_i. It is translated by the update operator $\langle \mathcal{M}'_i, w_i'^0 \cup \mathcal{M}'_{\circlearrowright}, w_{\circlearrowright}'^0 \rangle$ whose semantics is to choose *existentially* the update of the epistemic model by $\mathcal{M}'_i, w_i'^0$, that will give a new updated epistemic model with a chain of length i, that is p_i is true, or by $\mathcal{M}'_{\circlearrowright}, w_{\circlearrowright}'^0$, that will let the new updated epistemic model without a chain of length i, that is p_i is false. □

REMARK 1. *Note that the reduction used to prove that the model checking problem of $\mathcal{L}_{DEL}$ is PSPACE-hard uses only the precondition $\top$.*

4. SATISFIABILITY PROBLEM

The *satisfiability problem of* $\mathcal{L}_{DEL}$ is defined as follows:

Input: a formula $\varphi \in \mathcal{L}_{DEL}$;

Output: yes iff there exists a pointed epistemic model $(\mathcal{M}, w)$ such that $\mathcal{M}, w \models \varphi$.

The satisfiability problem is known to be decidable. Indeed, the standard reduction axioms of DEL [Baltag and Moss, 2004, p. 214] induce a translation $tr : \mathcal{L}_{DEL} \to \mathcal{L}_{EL}$ such that $\varphi \in \mathcal{L}_{DEL}$ is satisfiable iff $tr(\varphi) \in \mathcal{L}_{EL}$ is satisfiable. Since the size of $tr(\varphi)$ is at most exponential in the size of φ and the satisfiability problem of $\mathcal{L}_{EL}$ is PSPACE-complete, the satisfiability problem of $\mathcal{L}_{DEL}$ is in EXPSPACE. This upper bound is nevertheless not optimal in terms of complexity: we are going to prove in this section that the satisfiability problem of $\mathcal{L}_{DEL}$ is NEXPTIME-complete.

4.1 Upper bound

In this subsection we present a tableau method that does not rely on reduction axioms and we prove that it provides a NEXPTIME procedure deciding the satisfiability problem.

4.1.1 Tableau method

Let $\mathfrak{Lab}$ be a countable set of labels designed to represent worlds of the epistemic model $(\mathcal{M}, w)$. Our tableau method manipulates terms that we call tableau terms and they are of the following kind:

- $(\sigma \;\; \mathcal{M}'_1, w'_1; \dots; \mathcal{M}'_i, w'_i \;\; \varphi)$ where $\sigma \in \mathfrak{Lab}$ is a symbol (that represents a world in the initial model) and for all $j \in \{1, \dots, i\}$, $\mathcal{M}'_j, w'_j$ is an event model. This term means that φ is true in the world denoted by σ after the execution of the sequence $\mathcal{M}'_1, w'_1, \dots, \mathcal{M}'_i, w'_i$ and that the sequence is executable in the world denoted by σ;
- $(\sigma \;\; \mathcal{M}'_1, w'_1; \dots; \mathcal{M}'_i, w'_i \;\; \checkmark)$ means that the sequence $\mathcal{M}'_1, w'_1, \dots, \mathcal{M}'_i, w'_i$ is executable in the world denoted by σ;
- $(\sigma \;\; \mathcal{M}'_1, w'_1; \dots; \mathcal{M}'_i, w'_i \;\; \otimes)$ means that the sequence $\mathcal{M}'_1, w'_1, \dots, \mathcal{M}'_i, w'_i$ is not executable in the world denoted by σ;
- $(\sigma R_a \sigma_1)$ means that the world denoted by σ is linked by R_a to the world denoted by σ_1;
- $\bot$ denotes an inconsistency.

A *tableau rule* is represented by a *numerator* $\mathcal{N}$ above a line and a finite list of *denominators* $\mathcal{D}_1, \dots, \mathcal{D}_k$ below this line, separated by vertical bars:

$$\frac{\mathcal{N}}{\mathcal{D}_1 \mid \dots \mid \mathcal{D}_k}$$

$$\frac{(\sigma\ \Sigma'\ \varphi \wedge \psi)}{\begin{array}{c}(\sigma\ \Sigma'\ \varphi)\\ (\sigma\ \Sigma'\ \psi)\end{array}}\ (\wedge) \qquad \frac{(\sigma\ \Sigma'\ \neg\neg\varphi)}{(\sigma\ \Sigma'\ \varphi)}\ (\neg\neg)$$

$$\frac{(\sigma\ \Sigma'\ \neg(\varphi \wedge \psi))}{(\sigma\ \Sigma'\ \neg\varphi) \mid (\sigma\ \Sigma'\ \neg\psi)}\ (\neg\wedge) \qquad \frac{(\sigma\ \Sigma'\ p)(\sigma\ \Sigma'\ \neg p)}{\perp}\ (\perp)$$

$$\frac{(\sigma\ \Sigma'\ p)}{(\sigma\ \epsilon\ p)}\ (\leftarrow_p) \qquad \frac{(\sigma\ \Sigma'\ \neg p)}{(\sigma\ \epsilon\ \neg p)}\ (\leftarrow_{\neg p})$$

$$\frac{\begin{array}{l}(\sigma\ \ \mathcal{M}'_1, w'_1; \ldots ; \mathcal{M}'_i, w'_i\ \ B_a\varphi)\\ (\sigma\ R_a\ \sigma_1)\end{array}}{\begin{array}{l}(\sigma_1\ \ \mathcal{M}'_1, u'_1; \ldots ; \mathcal{M}'_i, u'_i\ \ \checkmark)\\ (\sigma_1\ \ \mathcal{M}'_1, u'_1; \ldots ; \mathcal{M}'_i, u'_i\ \ \varphi)\end{array} \Bigg| \ (\sigma_1\ \ \mathcal{M}'_1, u'_1; \ldots ; \mathcal{M}'_i, u'_i\ \ \otimes)}\ (B_a)$$

$$\frac{(\sigma\ \ \mathcal{M}'_1, w'_1; \ldots ; \mathcal{M}'_i, w'_i\ \ \neg B_a\varphi)}{\begin{array}{l}(\sigma\ R_a\ \sigma_{\text{new}})\\ (\sigma_{\text{new}}\ \ \mathcal{M}'_1, u'_1; \ldots ; \mathcal{M}'_i, u'_i\ \ \checkmark)\\ (\sigma_{\text{new}}\ \ \mathcal{M}'_1, u'_1; \ldots ; \mathcal{M}'_i, u'_i\ \ \neg\varphi)\end{array}}\ (\neg B_a)$$

$$\frac{(\sigma\ \Sigma'\ \neg[\mathcal{M}', w']\varphi)}{\begin{array}{l}(\sigma\ \Sigma'\ ; \mathcal{M}', w'\ \ \checkmark)\\ (\sigma\ \Sigma'\ ; \mathcal{M}', w'\ \ \neg\varphi)\end{array}}\ (\neg[\mathcal{M}', w'])$$

$$\frac{(\sigma\ \Sigma'\ [\mathcal{M}', w']\varphi)}{(\sigma\ \Sigma'\ ; \mathcal{M}', w'\ \ \otimes) \ \Bigg| \begin{array}{l}(\sigma\ \Sigma'\ ; \mathcal{M}', w'\ \ \checkmark)\\ (\sigma\ \Sigma'\ ; \mathcal{M}', w'\ \ \neg\varphi)\end{array}}\ ([\mathcal{M}', w'])$$

$$\frac{(\sigma\ \Sigma'\ ; \mathcal{M}', w'\ \ \checkmark)}{\begin{array}{l}(\sigma\ \Sigma'\ Pre(w'))\\ (\sigma\ \Sigma'\ \checkmark)\end{array}}\ (\checkmark) \qquad \frac{(\sigma\ \Sigma'\ ; \mathcal{M}', w'\ \ \otimes)}{\begin{array}{l}(\sigma\ \Sigma'\ \checkmark)\\ (\sigma\ \Sigma'\ \neg Pre(w'))\end{array} \Bigg| \ (\sigma\ \Sigma'\ \otimes)}\ (\otimes)$$

$$\frac{(\sigma\ \Sigma'\ \otimes)(\sigma\ \Sigma'\ \checkmark)}{\perp}\ (clash_{\checkmark,\otimes}) \qquad \frac{(\sigma\ \epsilon\ \otimes)}{\perp}\ (\epsilon_\otimes)$$

$$\frac{(\sigma\ \Sigma'\ [\pi \cup \gamma]\varphi)}{\begin{array}{l}(\sigma\ \Sigma'\ [\pi]\varphi)\\ (\sigma\ \Sigma'\ [\gamma]\varphi)\end{array}}\ ([\pi \cup \gamma]) \qquad \frac{(\sigma\ \Sigma'\ \neg[\pi \cup \gamma]\varphi)}{\begin{array}{l}(\sigma\ \Sigma'\ \neg[\pi]\varphi) \mid\\ (\sigma\ \Sigma'\ \neg[\gamma]\varphi)\end{array}}\ (\neg[\pi \cup \gamma])$$

Figure 4: Tableau rules

The numerator and the denominators are finite sets of tableau terms.

A *tableau tree* is a finite tree with a set of tableau terms at each node. A rule with numerator $\mathcal{N}$ is *applicable* to a node carrying a set Γ if Γ contains an instance of $\mathcal{N}$. If no rule is applicable, Γ is said to be *saturated*. We call a node σ an *end node* if the set of formulas Γ it carries is saturated, or if $\perp \in \Gamma$. The tableau tree is extended as follows:

1. Choose a leaf node n carrying Γ where n is not an end node, and choose a rule ρ applicable to n.
2. (a) If ρ has only one denominator, add the appropriate instanciation to Γ.

   (b) If ρ has k denominators with $k > 1$, create k successor nodes for n, where each successor i carries the union of Γ with an appropriate instanciation of denominator $\mathcal{D}_i$.

A branch in a tableau tree is a path from the root to an end node. A branch is *closed* if its end node contains $\perp$, otherwise it is *open*. A tableau tree is *closed* if all its branches are closed, otherwise it is *open*. The *tableau tree for a formula* $\varphi \in \mathcal{L}_{DEL}$ is the tableau tree obtained from the root $\{(\sigma_0\ \epsilon\ \varphi)\}$ when all leafs are end nodes. We write $\vdash \varphi$ when the tableau for $\neg\varphi$ is closed.

The tableau rules of our tableau method are represented in Figure 4. In these rules, Σ' is a list of pointed event models $\mathcal{M}'_1, w'_1, \ldots, \mathcal{M}'_i, w'_i$ and ϵ is the empty list. The tableau method contains the classical Boolean rules $(\wedge)$, $(\neg\neg)$, $(\leftarrow_p)$ and $(\leftarrow_{\neg p})$. It also contains the non-deterministic rule $(\neg\wedge)$ handling disjunction. The rule $(\perp)$ makes the current execution fail. The rule for (B_a) is applied for all $j \in \{1, \ldots i\}$ and all u'_j such that $w'_j R'_a u'_j$. Similarly, the rule for $(\neg B_a)$ is applied by choosing non-deterministically for all $j \in \{1, \ldots i\}$ some u'_j such that $w'_j R'_a u'_j$ and creating a new fresh label σ_{new}. The rules $(\checkmark)$, $(\otimes)$, $(clash_{\checkmark,\otimes})$ and $(\epsilon_\otimes)$ handle the preconditions. The last two rules $([\pi \cup \gamma])$ and $(\neg[\pi \cup \gamma])$ handle the union operator.

THEOREM 3 (SOUNDNESS AND COMPLETENESS). *Let* $\varphi \in \mathcal{L}_{DEL}$. *It holds that* $\vdash \varphi$ *iff* $\models \varphi$.

EXAMPLE 4. *We prove with our tableau method that the formula* $\varphi = \neg[\mathcal{M}'_1, w'_1][\mathcal{M}'_2, w'_2]B_2 B_1 B_2 p$ *from Example 3 is satisfiable, where* $\mathcal{M}'_1, w'_1$ *and* $\mathcal{M}'_2, w'_2$ *are defined in Example 2. In Figure 5, an open branch of the tableau tree for* φ *is represented. The set* Σ_{22} *is saturated: no more tableau rule is applicable. From this branch, we may extract a pointed epistemic model* $(\mathcal{M}, \sigma_0)$ *such that* $\mathcal{M}, \sigma_0 \models \varphi$.

4.1.2 NEXPTIME-membership

THEOREM 4. *The satisfiability problem of* $\mathcal{L}_{DEL}$ *is in NEXPTIME.*

PROOF SKETCH. Termination of our tableau method is proved by defining the size of a term $(\sigma\ \Sigma'\ \varphi)$ by $1 + \sum_{(\mathcal{M}', w') \in \Sigma'} (|\mathcal{M}'| + 1) + |\varphi|$. The depth of the tableau tree is linear in the size of the input formula, but the number of tableau terms at a node σ may be exponential, because of rule $(\neg B_a)$. As a consequence, the tableau tree has at most an exponential number of nodes and constructing non-deterministically such a tree can been done in an exponential amount of time. Hence, the procedure is in NEXPTIME. $\square$

4.2 Lower bound

$$\Sigma_0 := \{(\sigma_0 \; \epsilon \; \varphi)\}$$

$$\Big\downarrow \; {\scriptstyle (\neg[\mathcal{M}', w'])}$$

$$\Sigma_1 := \Sigma_0 \cup \left\{ \begin{array}{l} (\sigma_0 \; \mathcal{M}'_1, w'_1 \; \checkmark) \\ (\sigma_0 \; \mathcal{M}'_1, w'_1 \neg[\mathcal{M}'_2, w'_2] B_2 B_1 B_2 p) \end{array} \right\}$$

$$\Big\downarrow \; {\scriptstyle (\checkmark)}$$

$$\Sigma_2 := \Sigma_1 \cup \{ \; (\sigma_0 \; \epsilon \; \checkmark), (\sigma_0 \; \epsilon \; p) \; \}$$

$$\Big\downarrow \; {\scriptstyle (\neg[\mathcal{M}', w'])}$$

$$\Sigma_3 := \Sigma_2 \cup \left\{ \begin{array}{l} (\sigma_0 \; \mathcal{M}'_1, w'_1; \mathcal{M}'_2, w'_2 \; \checkmark) \\ (\sigma_0 \; \mathcal{M}'_1, w'_1; \mathcal{M}'_2, w'_2 \; \neg B_2 B_1 B_2 p) \end{array} \right\}$$

$$\Big\downarrow \; {\scriptstyle (\checkmark)}$$

$$\Sigma_4 := \Sigma_3 \cup \{ \; (\sigma_0 \; \mathcal{M}'_1, w'_1 \; \checkmark), (\sigma_0 \; \mathcal{M}'_1, w'_1 \; B_2 p) \; \}$$

$$\Big\downarrow \; {\scriptstyle (\neg B_a)}$$

$$\Sigma_5 := \Sigma_4 \cup \left\{ \begin{array}{l} (\sigma_0 \; R_2 \; \sigma_1) \\ (\sigma_1 \; \mathcal{M}'_1, w'_1; \mathcal{M}'_2, u'_2 \; \checkmark) \\ (\sigma_1 \; \mathcal{M}'_1, w'_1; \mathcal{M}'_2, u'_2 \; \neg B_1 B_2 p) \end{array} \right\}$$

$$\Big\downarrow \; {\scriptstyle (B_a)}$$

$$\Sigma_6 := \Sigma_5 \cup \{ \; (\sigma_1 \; \mathcal{M}'_1, w'_1 \; \checkmark), (\sigma_1 \; \mathcal{M}'_1, w'_1 \; p) \; \}$$

$$\Big\downarrow \; {\scriptstyle (\checkmark)}$$

$$\Sigma_7 := \Sigma_6 \cup \{ \; (\sigma_1 \; \mathcal{M}'_1, w'_1 \; \checkmark), (\sigma_1 \; \mathcal{M}'_1, w'_1 \; \neg(p \wedge \neg p)) \; \}$$

$$\Big\downarrow \; {\scriptstyle (\neg\wedge, \neg\neg)}$$

$$\Sigma_8 := \Sigma_7 \cup \{ \; (\sigma_1 \; \mathcal{M}'_1, w'_1 \; p) \; \}$$

$$\Big\downarrow \; {\scriptstyle (\rightarrow_p)}$$

$$\Sigma_9 := \Sigma_8 \cup \{ \; (\sigma_1 \; \epsilon \; p) \; \}$$

$$\Big\downarrow \; {\scriptstyle (\neg B_a)}$$

$$\Sigma_{10} := \Sigma_9 \cup \left\{ \begin{array}{l} (\sigma_1 \; R_1 \; \sigma_2) \\ (\sigma_2 \; \mathcal{M}'_1, u'_1; \mathcal{M}'_2, u'_2 \; \checkmark) \\ (\sigma_2 \; \mathcal{M}'_1, u'_1; \mathcal{M}'_2, u'_2 \; \neg B_2 p) \end{array} \right\}$$

$$\Big\downarrow \; {\scriptstyle (\checkmark)}$$

$$\Sigma_{11} := \Sigma_{10} \cup \{ \; (\sigma_2 \; \mathcal{M}'_1, u'_1 \; \checkmark), (\sigma_2 \; \mathcal{M}'_1, u'_1 \; \neg(p \wedge \neg p)) \; \}$$

$$\Big\downarrow \; {\scriptstyle (\neg\wedge, \neg\neg)}$$

$$\Sigma_{12} := \Sigma_{11} \cup \{ \; (\sigma_2 \; \mathcal{M}'_1, u'_1 \; p) \; \}$$

$$\Big\downarrow \; {\scriptstyle (\rightarrow_p)}$$

$$\Sigma_{13} := \Sigma_{12} \cup \{ \; (\sigma_2 \; \epsilon \; p) \; \}$$

$$\Big\downarrow \; {\scriptstyle (\checkmark)}$$

$$\Sigma_{14} := \Sigma_{13} \cup \{ \; (\sigma_2 \; \epsilon \; \checkmark), (\sigma_2 \; \epsilon \; \top) \; \}$$

$$\Big\downarrow \; {\scriptstyle (\neg\wedge, \neg\neg)}$$

$$\Sigma_{15} := \Sigma_{14} \cup \{ \; (\sigma_2 \; \epsilon \; p) \; \}$$

$$\Big\downarrow \; {\scriptstyle (\neg B_a)}$$

$$\Sigma_{16} := \Sigma_{15} \cup \left\{ \begin{array}{l} (\sigma_2 \; R_2 \; \sigma_3) \\ (\sigma_3 \; \mathcal{M}'_1, u'_1; \mathcal{M}'_2, u'_2 \; \checkmark) \\ (\sigma_3 \; \mathcal{M}'_1, u'_1; \mathcal{M}'_2, u'_2 \; \neg p) \end{array} \right\}$$

$$\Big\downarrow \; {\scriptstyle (\rightarrow_{\neg p})}$$

$$\Sigma_{17} := \Sigma_{16} \cup \{ \; (\sigma_3 \; \epsilon \; \neg p) \; \}$$

$$\Big\downarrow \; {\scriptstyle (\checkmark)}$$

$$\Sigma_{18} := \Sigma_{17} \cup \left\{ \begin{array}{l} (\sigma_3 \; \mathcal{M}'_1, u'_1 \; \checkmark) \\ (\sigma_3 \; \mathcal{M}'_1, u'_1; \mathcal{M}'_2, u'_2 \; \neg(p \wedge \neg p)) \end{array} \right\}$$

$$\Big\downarrow \; {\scriptstyle (\checkmark)}$$

$$\Sigma_{19} := \Sigma_{18} \cup \{ \; (\sigma_3 \; \epsilon \; \neg(p \wedge \neg p) \; \}$$

$$\Big\downarrow \; {\scriptstyle (\neg\wedge, \neg\neg)}$$

$$\Sigma_{20} := \Sigma_{19} \cup \{ \; (\sigma_3 \; \epsilon \; \neg p) \; \}$$

$$\Big\downarrow \; {\scriptstyle (\neg\wedge, \neg\neg)}$$

$$\Sigma_{21} := \Sigma_{20} \cup \{ \; (\sigma_3 \; \mathcal{M}'_1, u'_1; \mathcal{M}'_2, u'_2 \; \neg p) \; \}$$

$$\Big\downarrow \; {\scriptstyle (\rightarrow_{\neg p})}$$

$$\Sigma_{22} := \Sigma_{21} \cup \{ \; (\sigma_3 \; \epsilon \; \neg p) \; \}$$

Figure 5: An open branch of the tableau for φ

We prove that the algorithm based on our tableau method of the previous section is optimal in terms of computational complexity. To do so, we prove that the satisfiability problem of $\mathcal{L}_{DEL}$ is NEXPTIME-hard by reducing a NEXPTIME-complete tiling problem to it [Boas, 1997].

Let C be a countable and infinite set of colors. A *tile type* t is a 4-tuple of colors, denoted $t = (\mathit{left}(t), \mathit{right}(t), \mathit{up}(t), \mathit{down}(t)) \in C^4$. We consider the following *tiling problem*:

> **Input**: a finite set T of tile types, $t_0 \in T$ and a natural number k written in its binary form.
>
> **Output**: yes iff there exists a function τ from $\{0, \ldots k\}^2$ to T satisfying the following constraints:
>
> $$\tau(0,0) = t_0; \tag{1}$$
>
> for all $x \in \{0, \ldots, k\}$ and $y \in \{0, \ldots, k-1\}$:
>
> $$\mathit{up}(\tau(x,y)) = \mathit{down}(\tau(x, y+1)); \tag{2}$$
>
> for all $x \in \{0, \ldots, k-1\}$ and $y \in \{0, \ldots, k\}$:
>
> $$\mathit{right}(\tau(x,y)) = \mathit{left}(\tau(x+1, y)). \tag{3}$$

In other words, the problem is to decide whether we can tile a $k \times k$ grid with the tile types of T, t_0 being placed onto $(0,0)$.

Theorem 5. *The satisfiability problem of $\mathcal{L}_{DEL}$ is NEXPTIME -hard.*

Proof. Without loss of generality, we assume that $k = 2^n$. Let us consider an instance of the NEXPTIME-hard tiling problem described above. Our goal is to provide a polynomial translation from this instance to an instance of the satisfiability problem of $\mathcal{L}_{DEL}$.

The idea is to embed *two identical* $k \times k$-tilings into a single tree. Each leaf of the tree represents both a position (x_1, y_1) in the first tiling and a position (x_2, y_2) in the second tiling. We need to encode *two identical* tilings because, in order to check constraints 2 and 3, we will need to refer to the tile located to the right or to the left of a given position in a tiling, and also to refer to the tile located above or below it. This is hardly possible if we encode a single tiling at the leafs of a tree, because we would need to 'backtrack' in the tree to access these other positions.

We start by showing how to encode two identical tilings at the leafs of a tree. Then we will show how to express the three constraints 1, 2 and 3 in the definition of a tiling.

1. The coordinates (x_1, y_1) and (x_2, y_2) of the two tilings are represented by the valuations of atomic propositions $p_0, \ldots, p_{4n-1}$. More precisely, the set $X_1 = \{p_0, \ldots, p_{n-1}\}$ contains the atomic propositions encoding the binary representation of the integer x_1, $Y_1 = \{p_n, \ldots, p_{2n-1}\}$ contains the atomic propositions encoding the binary representation of the integer y_1, $X_2 = \{p_{2n}, \ldots, p_{3n-1}\}$ contains the atomic propositions encoding the binary representation of the integer x_2, and $Y_2 = \{p_{3n}, \ldots, p_{4n-1}\}$ contains the atomic propositions encoding the binary representation of the integer y_2. For instance, for $n = 4$, the coordinates $(x_1, y_1) = (4, 3)$ and $(x_2, y_2) = (11, 2)$ are represented at a leaf of the tree by the following valuation. We recall that in binary notation, 4 is represented by $\overline{100}$, 3 is represented by $\overline{11}$, 12 is represented by $\overline{1100}$ and 2 is represented by $\overline{10}$.

$$\underbrace{\neg p_0, p_1, \neg p_2, \neg p_3}_{4}\ \underbrace{\neg p_4, \neg p_5, p_6, p_7}_{3}$$

$$\underbrace{p_8, p_9, \neg p_{10}, \neg p_{11}}_{12}\ \underbrace{\neg p_{12}, \neg p_{13}, p_{14}, \neg p_{15}}_{2}$$

We then encode the existence of all valuations over $X_1 \cup Y_1 \cup X_2 \cup Y_2$ with the following formula:

$$\bigwedge_{l<4n} B_a^l \Bigg(\langle B_a \rangle p_l \wedge \langle B_a \rangle \neg p_l \wedge \bigwedge_{i<l} \left((p_i \rightarrow B_a p_i) \wedge (\neg p_i \rightarrow B_a \neg p_i) \right) \Bigg). \quad (4)$$

Formula 4 is true at a pointed epistemic model iff this pointed epistemic model is bisimilar up to modal depth $4n$ to a binary tree of depth $4n$ whose leafs contain all the possible valuations associated to $p_0, \ldots, p_{4n-1}$.

In order to check Constraints 2 and 3 in the definition of a tiling, we will need to refer to the tile located to the right or to the left of a given position in a tiling, and also to refer to the tile located above or below it. The following formulas encode the fact that any pair of coordinates (x_1, x_2) and (y_1, y_2) of the two tilings satisfy the properties $x_1 = x_2$, $x_1 = x_2 + 1$, $y_1 = y_2$ and $y_1 = y_2 + 1$ respectively:

$$(x_1 = x_2) \triangleq \bigwedge_{i<n} (p_i \leftrightarrow p_{i+2n}) \quad (5)$$

$$(y_1 = y_2) \triangleq \bigwedge_{n \leq i < 2n} (p_i \leftrightarrow p_{i+2n}) \quad (6)$$

$$(x_1 = x_2 + 1) \triangleq \bigvee_{i<n} \Bigg(\bigwedge_{j<i} (p_{j+2n} \leftrightarrow p_j) \wedge \neg p_{i+2n} \wedge p_i \wedge \bigwedge_{i<j<n} (p_{j+2n} \wedge \neg p_j) \Bigg) \quad (7)$$

$$(y_1 = y_2 + 1) \triangleq \bigvee_{n \leq i < 2n} \Bigg(\bigwedge_{n \leq j < i} (p_{j+2n} \leftrightarrow p_j) \wedge \neg p_{i+2n} \wedge p_i \wedge \bigwedge_{i<j<2n} (p_{j+2n} \wedge \neg p_j) \Bigg) \quad (8)$$

The tile types of the first tiling are represented by atomic propositions 1_t and the tile types of the second tiling are represented by atomic propositions $2_{t'}$, where t and t' range over T. They hold at a leaf of the tree whose coordinates correspond to (x_1, y_1) and (x_2, y_2) when the tile type of the first tiling at coordinate (x_1, y_1) is t and the tile type of the second tiling at coordinate (x_2, y_2) is t'.

Formulas 9 and 10 below encode the fact that, *at each leaf of the tree*, there is exactly one tile type for the first tiling and exactly one tile type for the second tiling. Formula 11 below encodes the fact that when these two pairs of coordinates coincide, that is when $x_1 = x_2$ and $y_1 = y_2$, then the tile type of the first tiling and the tile type of the second tiling are identical.

$$B_a^{4n} \left(\bigvee_{t \in T} 1_t \wedge \bigvee_{t \in T} 2_t \right) \quad (9)$$

$$B_a^{4n} \bigwedge \left\{ (1_t \rightarrow \neg 1_{t'}) \wedge (2_t \rightarrow \neg 2_{t'}) \mid t, t' \in T, t \neq t' \right\} \quad (10)$$

$$B_a^{4n} \left((x_1 = x_2) \wedge (y_1 = y_2) \rightarrow \bigwedge_{t \in T} (1_t \leftrightarrow 2_t) \right) \quad (11)$$

However, it may be the case that in the tree, two different leafs with the *same* valuation have different tile types. Therefore, we also have to constrain the tree so that the leafs denoting the same position in the first tiling (resp. second tiling) contain the same tile type for the first tiling (resp. second tiling). This is expressed by the following two formulas:

$$[\mathcal{M}'_{p_0} \cup \mathcal{M}'_{\neg p_0}] \ldots [\mathcal{M}'_{p_{2n-1}} \cup \mathcal{M}'_{\neg p_{2n-1}}] \bigvee_{t \in T} B_a^{4n} 1_t \quad (12)$$

$$[\mathcal{M}'_{p_{2n}} \cup \mathcal{M}'_{\neg p_{2n}}] \ldots [\mathcal{M}'_{p_{4n-1}} \cup \mathcal{M}'_{\neg p_{4n-1}}] \bigvee_{t \in T} B_a^{4n} 2_t \quad (13)$$

where for a given a literal ℓ (p or $\neg p$), the pointed event model $\mathcal{M}'_\ell = (W', R', Pre, w'_0)$ is defined as follows: $W' = \{w'_i \mid i \in \{0, \ldots, 4n\}\}$; $R'_a = \{(w'_i, w'_{i+1}) \mid i \in \{0, \ldots, 4n-1\}\}$; and $Pre(w'_i) = \top$ for all $i < 4n$ and $Pre(w'_{4n}) = \ell$.

In formula 12, the sequence of pointed event models $[\mathcal{M}'_{p_0} \cup \mathcal{M}'_{\neg p_0}] \ldots [\mathcal{M}'_{p_{2n-1}} \cup \mathcal{M}'_{\neg p_{2n-1}}]$ non-deterministically picks a valuation v over $X_1 \cup Y_1$ and selects the branches of the tree whose leafs satisfy this valuation. Then, the formula $\bigvee_{t \in T} B_a^{4n} 1_t$ checks that these leafs, which denote the same position in the first tiling, are of the same tile type t. Likewise with formula 13 for the second tiling.

So, with formulas 9, 10, 11, 12 and 13, we have encoded in the tree two identical tilings in a single tree. Importantly, note that the tree is defined so that each leaf refers to two coordinates of the tiling, which can possibly be identical or consecutive. It is this feature which will allow us to express that constraints 2 and 3 of the definition of a tiling hold.

2. Constraints 1, 2 and 3 of the definition of a tiling are expressed respectively by the following formulas:

$$B_a^{4n} \left(\left(\bigwedge_{i<4n} \neg p_i \right) \rightarrow t_0 \right) \quad (14)$$

$$B_a^{4n} \Bigg((x_1 = x_2) \wedge (y_1 = y_2 + 1) \rightarrow \bigwedge_{t \in T} \Big\{ 1_t \rightarrow \bigvee \left\{ 2_{t'} \mid t' \in T, down(t') = up(t) \right\} \Big\} \Bigg) \quad (15)$$

$$B_a^{4n}\Bigg((x_1 = x_2 + 1) \wedge (y_1 = y_2) \qquad (16)$$
$$\rightarrow \bigwedge_{t \in T} \Big\{1_t \rightarrow \bigvee \{2_{t'} \mid t' \in T, left(t') = right(t)\}\Big\}\Bigg)$$

As we said at the begining of the proof, these two constraints motivate the need to encode *two* tilings: for a given position in a tiling, we need to refer to the tile located to the right or to the left of it, and to refer to the tile located above or below it. This would not be possible with our epistemic language if the tiling was encoded by a single tree.

One can then check that there exists a tiling for the instance of the tiling problem iff the formula φ, which is the conjunction of fomulas 4, 9, 10, 11, 12, 13, 14, 15, and 16 is satisfiable in $\mathcal{L}_{DEL}$.

3. Finally, we show that the reduction is polynomial in the size of the instance of the tiling problem. The formula of Equation 4 is of size $O(n^2)$. The formulas of Equations 12, 13 are of size $O(n^2 + |T| \times n)$. The other formulas are clearly of size polynomial in the size of the input, so the result follows. Importantly, note that if we decided to rewrite the formulas 12 and 13 without using the union operator $\cup$, then the corresponding formula would be exponential in the size of the input. So, the use of the union operator is really crucial in order to have a polynomial reduction from the tiling problem to our satisfiability problem. $\square$

5. RELATED WORK

5.1 Theory

There exists a terminating tableau method solving the satisfiability problem of $\mathcal{L}_{DEL}$ [Hansen, 2010]. This method writes subformulas by using the reduction axioms [Baltag and Moss, 2004, p. 214]. It is therefore mainly a variant of the tableau method of classical multi-modal logic K_n. Even if we know that *tr* blows up exponentially the size of the input formula, the computational complexity of this tableau method is not studied. In this section, we review the existing results about computational complexity of DEL.

5.1.1 Public Announcement Logic (PAL)

Public Announcement Logic (PAL) [Plaza, 1989] is an extension of epistemic logic with a dynamic operator $[\psi!]\varphi$ whose truth conditions are defined as follows:

$$\mathcal{M}, w \models [\psi!]\varphi \quad \text{iff} \quad \mathcal{M}, w \models \psi \text{ implies } \mathcal{M}_\psi, w \models \varphi$$

where $\mathcal{M}_\psi$ is the restriction of $\mathcal{M}$ to the worlds which satisfy ψ. PAL is a fragment of DEL: the language of PAL is $\mathcal{L}_{DEL}$ restricted to event models consisting of a single possible event with reflexive arrows for all agents. There is a gap between PAL and DEL in terms of computational complexity, both for the model checking problem and the satisfiability problem. Indeed, the model checking of PAL is in P (also with common belief) [van Benthem and Kooi, 2004] and the satisfiability problem for PAL is PSPACE-complete [Lutz, 2006]. Despite the fact that there exist reduction axioms for PAL, it is difficult to implement a direct translation using reduction axioms. In fact, there are properties that can be expressed exponentially more succinctly in PAL than in epistemic logic. Note that there exist PSPACE tableau methods for solving the satisfiability problem in public announcement logic [de Boer, 2007, Balbiani et al., 2010].

5.1.2 DEL-sequents

DEL-sequents [Aucher, 2011] are triples of the form $\varphi, \varphi' \models \varphi''$ where $\varphi, \varphi'' \in \mathcal{L}_{EL}$ and φ' is a formula of a language for event models. A DEL-sequent $\varphi, \varphi' \models \varphi''$ holds when for all pointed epistemic model $(\mathcal{M}, w)$ such that $\mathcal{M}, w \models \varphi$, for all pointed event model $(\mathcal{M}', w')$ such that $\mathcal{M}', w' \models \varphi'$, if $(\mathcal{M}', w')$ is executable in $(\mathcal{M}, w)$, then $\mathcal{M} \otimes \mathcal{M}', (w, w') \models \varphi''$. The problem of determining whether a DEL-sequent holds is NEXPTIME-complete and there exists a tableau method for it. DEL-sequents have been generalized to sequences of the form $\varphi_0, \varphi'_1, \varphi_1, \ldots, \varphi'_n, \varphi_n \models^1_i \psi$ and $\varphi_0, \varphi'_1, \varphi_1, \ldots, \varphi'_n, \varphi_n \models^2_i \psi'$. The corresponding satisfiability problem is also NEXPTIME-complete [Aucher et al., 2012].

5.1.3 The sequence and 'star' iteration operators

The sequence and 'star' iteration operators are constructions enabling to build complex programs as in Propositional Dynamic Logic (PDL [Harel et al., 2000]). The truth conditions are defined as follows:

$$\begin{array}{lll} \mathcal{M}, w \models [\pi;\gamma]\varphi & \text{iff} & \mathcal{M}, w \models [\pi][\gamma]\varphi \\ \mathcal{M}, w \models [\pi^*]\varphi & \text{iff} & \text{there is a } \textit{finite} \text{ sequence } \pi; \ldots; \pi \\ & & \text{such that } \mathcal{M}, w \models [\pi; \ldots; \pi]\varphi \end{array}$$

We do not know about the computational complexity of the model-checking problem when the operator $[\pi^*]\varphi$ is added to the language. In fact, we do not even know whether it is decidable. The computational complexity of the satisfiability problem remains the same when the sequential composition operator is added. However, adding a 'star' operator makes the satisfiability problem undecidable. This result is not really surprising, it is a direct corollary of the result of [Miller and Moss, 2005] stating that Public Announcement Logic with the 'star' operator is already undecidable.

5.1.4 The common belief operator

We may extend the language with the common knowledge operator $C_G\varphi$, where $G \subseteq AGT$. The truth conditions are defined as follows:

$$\mathcal{M}, w \models C_G\varphi \quad \text{iff} \quad \text{for all } v \in \Big(\bigcup_{a \in G} R_a\Big)^+(w), \mathcal{M}, v \models \varphi$$

Intuitively, $C_G\varphi$ is an abbreviation of an infinite conjunction [Fagin et al., 1995]: $C_G\varphi = E^1_G\varphi \wedge E^2_G\varphi \wedge E^3_G\varphi \wedge \ldots$, where $E^k_G\varphi$ is defined inductively as follows: $E^1_G\varphi = \bigwedge_{a \in G} B_a\varphi$ and $E^{k+1}_G\varphi = E^1_G E^k_G\varphi$.

We do not know about the computational complexity of the satisfiability problem when the common belief operator is added to the language $\mathcal{L}_{DEL}$. However, we know that it is decidable and that the language with common belief operator is more expressive than the *epistemic* language $\mathcal{L}_{EL}$ with common belief [Baltag et al., 1998, Baltag et al., 1999].

5.2 Implementation

There exist two implementations of our decision problems:

1. The model-checker DEMO [van Eijck, 2007], standing for Dynamic Epistemic MOdeling tool, can evaluate formulas of $\mathcal{L}_{DEL}$ in epistemic models, display graphically epistemic models, event models and updates of epistemic models by event models, translate formulas of $\mathcal{L}_{DEL}$ to formulas of PDL. DEMO is written in Haskel and has been applied in [van Ditmarsch et al., 2005] and [van Ditmarsch et al., 2006]. Also, it has been used to investigate the pros and cons of

modeling some well-known problems of computer security within the DEL framework [van Eijck and Orzan, 2007].

2. The program `Aximo` [Richards and Sadrzadeh, 2009], written in C++, implements an algorithm for proving properties of interactive multi-agent scenarios encoded in epistemic systems. Epistemic systems provide an algebraic semantics to DEL and were developed together with a sound and complete sequent calculus [Baltag et al., 2007].

6. CONCLUDING REMARKS

Our work contributes to the proof theory and the study of the computational complexity of DEL, which has been rather neglected so far. Although our results show that our decision problems are untractable, it turns out that the DEMO implementation does not fare worse and often even better in terms of time of execution than other model-checkers modeling the same problems without resorting to the DEL methodology [van Ditmarsch et al., 2006].

We still need to investigate whether or not the computational complexity remains the same when we consider other logics like S5, and how our tableau method can be extended to cover these other logics. We also need to investigate tractable fragments of our decision problems. Finally, we plan to implement our tableau method in LotrecScheme [Schwarzentruber, 2011].

7. REFERENCES

[Aucher, 2011] Aucher, G. (2011). DEL-sequents for progression. *Journal of Applied Non-Classical Logics*, 21(3-4):289–321.

[Aucher et al., 2012] Aucher, G., Maubert, B., and Schwarzentruber, F. (2012). Generalized DEL-sequents. In del Cerro, L. F., Herzig, A., and Mengin, J., editors, *JELIA*, volume 7519 of *Lecture Notes in Computer Science*, pages 54–66. Springer.

[Balbiani et al., 2010] Balbiani, P., van Ditmarsch, H., Herzig, A., and de Lima, T. (2010). Tableaux for public announcement logic. *Journal of Logic and Computation*, 20(1):55–76.

[Baltag et al., 2007] Baltag, A., Coecke, B., and Sadrzadeh, M. (2007). Epistemic actions as resources. *Journal of Logic and Computation*, 17(3):555–585.

[Baltag and Moss, 2004] Baltag, A. and Moss, L. (2004). Logic for epistemic programs. *Synthese*, 139(2):165–224.

[Baltag et al., 1998] Baltag, A., Moss, L., and Solecki, S. (1998). The logic of common knowledge, public announcement, and private suspicions. In Gilboa, I., editor, *Proceedings of the 7th conference on theoretical aspects of rationality and knowledge (TARK98)*, pages 43–56.

[Baltag et al., 1999] Baltag, A., Moss, L., and Solecki, S. (1999). The logic of public announcements, common knowledge and private suspicions. Technical report, Indiana University.

[Boas, 1997] Boas, P. (1997). The convenience of tilings. In *Complexity, Logic, and Recursion Theory*, pages 331–363. Marcel Dekker Inc.

[de Boer, 2007] de Boer, M. (2007). KE tableaux for public anouncement logic. In *Proceedings of Formal Approaches to Multi-Agent Systems Workshop (FAMAS 07)*, Durham UK.

[Fagin et al., 1995] Fagin, R., Halpern, J., Moses, Y., and Vardi, M. (1995). *Reasoning about knowledge.* MIT Press.

[Fitting, 1983] Fitting, M. (1983). *Proof methods for modal and intuitionistic logics.* D. Reidel, Dordrecht.

[Hansen, 2010] Hansen, J. (2010). Terminating tableaux for dynamic epistemic logics. *Electronic Notes in Theoretical Computer Science*, 262:141–156.

[Harel et al., 2000] Harel, D., Kozen, D., and Tiuryn, J. (2000). *Dynamic Logic.* MIT Press.

[Lutz, 2006] Lutz, C. (2006). Complexity and succinctness of public announcement logic. In *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, pages 137–143. ACM.

[Miller and Moss, 2005] Miller, J. and Moss, L. (2005). The undecidability of iterated modal relativization. *Studia Logica*, 79(3):373–407.

[Papadimitriou, 1995] Papadimitriou, C. H. (1995). *Computational complexity.* Addison Wesley.

[Plaza, 1989] Plaza, J. (1989). Logics of public communcations. In Emrich, M. L., Pfeifer, M. Z., Hadzikadic, M., and Ras, Z. W., editors, *Proceedings of the 4th International Symposium on Methodologies for Intelligent Systems*, pages 201–216.

[Richards and Sadrzadeh, 2009] Richards, S. and Sadrzadeh, M. (2009). Aximo: Automated axiomatic reasoning for information update. *Electr. Notes Theor. Comput. Sci.*, 231:211–225.

[Schwarzentruber, 2011] Schwarzentruber, F. (2011). Lotrecscheme. *Electr. Notes Theor. Comput. Sci.*, 278:187–199.

[van Benthem and Kooi, 2004] van Benthem, J. and Kooi, B. (2004). Reduction axioms for epistemic actions. In Schmidt, R., Pratt-Hartmann, I., Reynolds, M., and Wansing, H., editors, *AiML-2004: Advances in Modal Logic*, number UMCS-04-9-1 in Technical Report Series, pages 197–211, University of Manchester.

[van Ditmarsch et al., 2007] van Ditmarsch, H., van der Hoek, W., and Kooi, B. (2007). *Dynamic Epistemic Logic*, volume 337 of *Synthese library.* Springer.

[van Ditmarsch et al., 2005] van Ditmarsch, H. P., Ruan, J., and Verbrugge, L. C. (2005). Model checking sum and product. In Zhang, S. and Jarvis, R., editors, *Australian Conference on Artificial Intelligence*, volume 3809 of *Lecture Notes in Computer Science*, pages 790–795. Springer.

[van Ditmarsch et al., 2006] van Ditmarsch, H. P., van der Hoek, W., van der Meyden, R., and Ruan, J. (2006). Model checking russian cards. *Electr. Notes Theor. Comput. Sci.*, 149(2):105–123.

[van Eijck, 2007] van Eijck, J. (2007). Demo — a demo of epistemic modelling. In van Benthem, J., Gabbay, D., and Löwe, B., editors, *Interactive Logic — Proceedings of the 7th Augustus de Morgan Workshop*, Texts in Logic and Games 1, pages 305–363.

[van Eijck and Orzan, 2007] van Eijck, J. and Orzan, S. (2007). Epistemic verification of anonymity. *Electronic Notes in Theoretical Computer Science*, 168(0):159 – 174. Proceedings of the Second International Workshop on Views on Designing Complex Architectures (VODCA 2006).

APPENDIX

A. PROOF OF THEOREM 1

THEOREM 6. *The model-checking problem of DEL is in PSPACE.*

PROOF. Terminaison and correction of the algorithm `M-Check` are easily proved over the size of the input defined by:

$$|\mathcal{M}| + \sum_{k=1}^{i} |\mathcal{M}'_k| + |\varphi|.$$

The only case worth mentionning is when φ is of the form $[\mathcal{M}', w']\psi$. In that case, the number of recursive calls of `M-Check`$(w \;\; \mathcal{M}'_1, w'_1; \ldots; \mathcal{M}'_i, w'_i; \mathcal{M}', w' \;\; \psi)$ is of size $|\mathcal{M}| + \sum_{k=1}^{i} |\mathcal{M}'_k| + |\mathcal{M}'| + |\psi|$. This quantity is indeed smaller than the size of the call of φ, because it is equal to $|\mathcal{M}| + \sum_{k=1}^{i} |\mathcal{M}'_k| + |\varphi| = |\mathcal{M}| + \sum_{k=1}^{i} |\mathcal{M}'_k| + |\mathcal{M}'| + |\psi| + 1$.

The algorithm requires a polynomial amount of space in the size of the input. Indeed, as the size of the input is strictly decreasing at each recursive call, the number of recursive calls in the call stack is linear in the size of the input. Then each of the current call requires a polynomial amount of space in the size of the input for storing the value of local variables: the most consuming case is $B_a\psi$ where we have to save all the current values of u, $u_1, \ldots, u_i$ in the loop **for**. $\square$

B. PROOF OF THEOREM 3

PROPOSITION 1 (SOUNDNESS). *Let $\varphi \in \mathcal{L}_{DEL}$. If $\vdash \varphi$, then $\models \varphi$.*

PROOF. We prove it by contrapositive. Suppose that $\not\models \varphi$, that is, there exists a pointed epistemic model $(\mathcal{M}, w)$ such that $\mathcal{M}, w \models \neg\varphi$. We must prove that every tableau for $(\sigma_0 \;\epsilon\; \neg\varphi)$ has an open branch (the proof of termination is in the proof of Theorem 4).

We say that a set Γ_f of tableau terms is *interpretable* if there exists a pointed epistemic model $\mathcal{M} = (W, R, V)$, a partial function $\nu : \mathfrak{Lab} \to W$ such that $(\mathcal{M}, \nu)$ makes all the tableau terms in Γ_f true for the following semantics $\models_T$:

$(\mathcal{M}, \nu) \models_T (\sigma \;\; \mathcal{M}'_1, w'_1; \ldots; \mathcal{M}'_i, w'_i \;\; \varphi)$
iff the sequence $(\mathcal{M}'_1, w'_1), \ldots, (\mathcal{M}'_i, w'_i)$ is executable in $(\mathcal{M}, \nu(\sigma))$ and $\mathcal{M} \otimes \mathcal{M}'_1 \otimes \ldots \otimes \mathcal{M}'_i, (\nu(\sigma), w'_1, \ldots, w'_i) \models \varphi$
$(\mathcal{M}, \nu) \models_T (\sigma \;\; \mathcal{M}'_1, w'_1; \ldots; \mathcal{M}'_i, w'_i \;\; \checkmark)$
iff the sequence $(\mathcal{M}'_1, w'_1), \ldots, (\mathcal{M}'_i, w'_i)$ is executable in $(\mathcal{M}, \nu(\sigma))$
$(\mathcal{M}, \nu) \models_T (\sigma \;\; \mathcal{M}'_1, w'_1; \ldots; \mathcal{M}'_i, w'_i \;\; \otimes)$
iff the sequence $(\mathcal{M}'_1, w'_1), \ldots, (\mathcal{M}'_i, w'_i)$ is *not* executable in $(\mathcal{M}, \nu(\sigma))$
$(\mathcal{M}, \nu) \models_T (\sigma \; R_a \; \sigma_1)$ iff $\nu(\sigma) \; R_a \; \nu(\sigma_1)$
$(\mathcal{M}, \nu) \models_T \bot$ iff false

Since $\not\models \varphi$, the set $\Gamma = \{(\sigma_0 \;\; \neg\varphi)\}$ is interpretable. Moreover, if a set of formulas is interpretable, it does not contain $\bot$. So, if we prove that when the numerator of a rule is interpretable, one of the denominators also is, then we have that every tableau for $(\sigma_0 \;\epsilon\; \neg\varphi)$ has an open branch. In that case, we say that the rule is *sound*. We only prove it for $(\neg B_a)$, (B_a), the proof for the other rules being similar. In the following, when ν is a function, we let $\nu(x \mapsto a)$ be the function that maps x to a and y to $\nu(y)$ if $y \neq x$.

- Rule $(\neg B_a)$: If $(\mathcal{M}, \nu) \models_T (\sigma \;\; \mathcal{M}'_1, w'_1; \ldots; \mathcal{M}'_i, w'_i \;\; \neg B_a\psi)$, then $\mathcal{M} \otimes \mathcal{M}'_1 \otimes \ldots \otimes \mathcal{M}'_i, (\nu(\sigma), w'_1, \ldots, w'_i) \models \neg B_a\psi$. So, there exists $(u, u'_1, \ldots, u'_i) \in R_a(\nu(\sigma), w'_1, \ldots, w'_i)$ such that $\mathcal{M} \otimes \mathcal{M}'_1 \otimes \ldots \otimes \mathcal{M}'_i, (u, u'_1, \ldots, u'_i) \models \neg\psi$. So, $u \in R_a(\nu(\sigma))$, the sequence $(\mathcal{M}'_1, u'_1), \ldots, (\mathcal{M}'_i, u'_i)$ is executable in $(\mathcal{M}, w)$ and $\mathcal{M} \otimes \mathcal{M}'_1 \otimes \ldots \otimes \mathcal{M}'_i, (u, u'_1, \ldots, u'_i) \models \neg\psi$. Let σ_{new} be a new fresh label and let $\nu^+ := \nu(\sigma_{\text{new}} \mapsto u)$. Then, we have that $(\mathcal{M}, \nu^+) \models_T (\sigma R_a \sigma_{\text{new}}), (\mathcal{M}, \nu^+) \models_T (\sigma_{\text{new}} \mathcal{M}'_1, u'_1; \ldots; \mathcal{M}'_i, u'_i \;\checkmark)$ and $(\mathcal{M}, \nu^+) \models_T (\sigma_{\text{new}} \;\; \mathcal{M}'_1, u'_1; \ldots; \mathcal{M}'_i, u'_i \;\; \checkmark \neg\psi)$. So, the rule $(\neg B_)$ is sound.

- Rule (B_a): If $(\mathcal{M}, \nu) \models_T (\sigma \;\; \mathcal{M}'_1, w'_1; \ldots; \mathcal{M}'_i, w'_i \;\; B_a\varphi)$ and $(\mathcal{M}, \nu) \models_T (\sigma \; R_a \; \sigma_1)$, then $\mathcal{M} \otimes \mathcal{M}'_1 \otimes \ldots \otimes \mathcal{M}'_i, (\nu(\sigma), w'_1, \ldots, w'_i) \models B_a\varphi$ and $\nu(\sigma) R_a \nu(\sigma_1)$. Let $u'_1 \in R_a(w'_1), \ldots, u'_i \in R_a(w'_i)$. Either the sequence $(\mathcal{M}_1, u'_1), \ldots, (\mathcal{M}'_i, u'_i)$ is executable in $(\mathcal{M}, \nu(\sigma_1))$ or it is not. In the second case, $(\mathcal{M}, \nu) \models_T (\sigma \;\; \mathcal{M}'_1, w'_1; \ldots; \mathcal{M}'_i, w'_i \;\; \otimes)$ by definition of $\models_T$ and the denominator is interpretable. If the sequence is executable, then $(\mathcal{M}, \nu) \models_T (\sigma_1 \;\; \mathcal{M}'_1, u'_1; \ldots; \mathcal{M}'_i, u'_i \;\; \checkmark)$ by definition of $\models_T$. Moreover, $\mathcal{M} \otimes \mathcal{M}'_1 \otimes \ldots \otimes \mathcal{M}'_i, (\nu(\sigma_1), u'_1, \ldots, u'_i) \models \varphi$, that is, $(\mathcal{M}, \nu) \models_T (\sigma_1 \;\; \mathcal{M}'_1, u'_1; \ldots; \mathcal{M}'_i, u'_i \;\; \varphi)$. So, in the first case, the denominator is also interpretable. Hence, the rule (B_a) is sound. $\square$

PROPOSITION 2 (COMPLETENESS). *Let $\varphi \in \mathcal{L}_{DEL}$. If $\models \varphi$, then $\vdash \varphi$.*

PROOF. We prove it by contrapositive. Suppose that there is a tableau for φ that has an open branch. We prove that there is a pointed epistemic model $(\mathcal{M}, w)$ such that $\mathcal{M}, w \models \varphi$. Let Γ_f be the set of tableau terms carried by the end node of the open branch. We define the pointed epistemic model $(\mathcal{M}, w) = (W, R, V, w)$ as follows:

- $W = \{w_\sigma \mid \sigma \in \mathfrak{Lab}$ appears in a tableau term of $\Gamma_f\}$;
- $R_a = \{(w_\sigma, w_\tau) \in W \times W \mid (\sigma R_a \tau) \in \Gamma_f\}$ for all $a \in AGT$;
- $V(p) = \{w_\sigma \in W \mid (\sigma \;\epsilon\; p) \in \Gamma_f\}$ for all $p \in ATM$;
- $w = w_{\sigma_0}$.

For all $n \in \mathbb{N}$, let $\mathcal{P}(n)$ be the induction hypothesis defined as follows: $\mathcal{P}(n)$="let $\varphi \in \mathcal{L}_{DEL}$ and let $(\mathcal{M}'_1, w'_1), \ldots, (\mathcal{M}'_i, w'_i)$ be i pointed event models such that $\sum_{k=1}^{i} |\mathcal{M}'_k| + |\varphi| = n$. If $(\sigma \;\; \mathcal{M}'_1, w'_1; \ldots; \mathcal{M}'_i, w'_i \;\; \varphi) \in \Gamma_f$ and $(\sigma \;\; \mathcal{M}'_1, w'_1; \ldots; \mathcal{M}'_i, w'_i \;\; \checkmark) \in \Gamma_f$, then the sequence $(\mathcal{M}'_1, w'_1), \ldots, (\mathcal{M}'_i, w'_i)$ is executable in $(\mathcal{M}, w_\sigma)$ and $\mathcal{M} \otimes \mathcal{M}'_1 \otimes \ldots \otimes \mathcal{M}'_i, (w_\sigma, w'_1, \ldots, w'_i) \models \varphi$".

$n = 1$. In that case, we necessarily have that $\varphi = p$ for some $p \in ATM$ and $i = 0$. So, $(\sigma \;\epsilon\; p) \in \Gamma_f$, and therefore $\mathcal{M}, w_\sigma \models p$ by definition of $\mathcal{M}$.

$n + 1$. If $i \neq 0$, then by saturation of rule $(\checkmark)$, we have that $(\sigma \;\; \mathcal{M}'_1, w'_1; \ldots; \mathcal{M}'_i, w'_i \;\; Pre(w')) \in \Gamma_f$ because $\sum_{k=1}^{i-1} |\mathcal{M}'_k| + |Pre(w')| < \sum_{k=1}^{i} |\mathcal{M}'_k| + |\varphi|$, we also have that the sequence $(\mathcal{M}'_1, w'_1), \ldots, (\mathcal{M}'_{i-1}, w'_{i-1})$ is executable in $(\mathcal{M}, w_\sigma)$ and $\mathcal{M} \otimes \mathcal{M}'_1 \otimes \ldots \otimes \mathcal{M}'_{i-1}, (w_\sigma, w'_1, \ldots, w'_{i-1}) \models Pre(w')$. Therefore, $(\mathcal{M}'_1, w'_1), \ldots, (\mathcal{M}'_i, w'_i)$ is executable in $(\mathcal{M}, w_\sigma)$. So, we have proved the first part of the induction step. Now, we prove the second part, namely that $\mathcal{M} \otimes \mathcal{M}'_1 \otimes \ldots \otimes \mathcal{M}'_i, (w_\sigma, w'_1, \ldots, w'_i) \models \varphi$. We only prove it for $i \neq 0$, the case $i = 0$ being similar. We reason by cases depending on the structure of φ. Note that we do not introduce another sub-reasoning by induction, we only reason by distinguishing sub-cases.

- If $\varphi = p$, then $(\sigma\ \epsilon\ p) \in \Gamma_f$ by saturation of $(\leftarrow_p)$. So, $\mathcal{M}, w_\sigma \models p$ by definition of $\mathcal{M}$. Hence, by definition of the product update, $\mathcal{M} \otimes \mathcal{M}'_1 \otimes \ldots \otimes \mathcal{M}'_i, (w_\sigma, w'_1, \ldots, w'_i) \models p$. The case $\varphi = \neg p$ is dealt with similarly. As for the case $\varphi = \varphi_1 \wedge \varphi_2$, by saturation of rule $(\wedge)$, Γ_f also contains the tableau term $(\sigma\ \ \mathcal{M}'_1, w'_1; \ldots; \mathcal{M}'_i, w'_i\ \ \varphi_1)$ and $(\sigma\ \ \mathcal{M}'_1, w'_1; \ldots; \mathcal{M}'_i, w'_i\ \ \varphi_2)$. Moreover, by saturation of rule $(\checkmark)$, Γ_f also contains $(\sigma\ \ \mathcal{M}'_1, w'_1; \ldots; \mathcal{M}'_i, w'_i\ \ \checkmark)$. So, by Induction Hypothesis, we have that $\mathcal{M} \otimes \mathcal{M}'_1 \otimes \ldots \otimes \mathcal{M}'_i, (w_\sigma, w'_1, \ldots, w'_i) \models \varphi_1$ and also $\mathcal{M} \otimes \mathcal{M}'_1 \otimes \ldots \otimes \mathcal{M}'_i, (w_\sigma, w'_1, \ldots, w'_i) \models \varphi_2$. So, $\mathcal{M} \otimes \mathcal{M}'_1 \otimes \ldots \otimes \mathcal{M}'_i, (w_\sigma, w'_1, \ldots, w'_i) \models \varphi$. The case $\varphi = \neg(\varphi_1 \wedge \varphi_2)$ and $\varphi = \neg\neg\psi$ are proved similarly.

- If $\varphi = \neg B_a \psi$, then by saturation of rule $(\neg B_a)$, we have the existence of σ_{new} such that $(\sigma R_a \sigma_{new}) \in \Gamma_f$ and the existence of $u'_1 \in R'_a(w'_1), \ldots, u'_i \in R'_a(w'_i)$ such that $(\sigma_{\text{new}}\ \ \mathcal{M}'_1, u'_1; \ldots; \mathcal{M}'_i, u'_i\ \ \checkmark) \in \Gamma_f$ and $(\sigma_{\text{new}}\ \ \mathcal{M}'_1, u'_1; \ldots; \mathcal{M}'_i, u'_i\ \ \neg\psi) \in \Gamma_f$. Now, $\sum_{k=1}^{i} |\mathcal{M}'_k| + |\neg\psi| < \sum_{k=1}^{i} |\mathcal{M}'_k| + |\varphi|$. So, by Induction Hypothesis, $(\mathcal{M}'_1, u'_1), \ldots, (\mathcal{M}'_i, u'_i)$ is executable in $(\mathcal{M}, w_\sigma)$ and $\mathcal{M} \otimes \mathcal{M}'_1 \otimes \ldots \otimes \mathcal{M}'_i, (w_\sigma, u'_1, \ldots u'_i) \models \neg\psi$. That is, $\mathcal{M} \otimes \mathcal{M}'_1 \otimes \ldots \otimes \mathcal{M}'_i, (w_\sigma, w'_1, \ldots, w'_i) \models \neg B_a \psi$.

- If $\varphi = B_a \psi$, then let us consider some $w_{\sigma_1} \in R_a(w_\sigma), u'_1 \in R'_a(w'_1), \ldots, u'_i \in R'_a(w'_i)$ such that $(w_{\sigma_1}, u'_1, \ldots, u'_i) \in \mathcal{M} \otimes \mathcal{M}'_1 \otimes \ldots \otimes \mathcal{M}'_i$. We are going to show that $\mathcal{M} \otimes \mathcal{M}'_1 \otimes \ldots \otimes \mathcal{M}'_i, (w_{\sigma_1} u'_1, \ldots, u'_i) \models \psi$. By definition of $\mathcal{M}$, $(\sigma R_a \sigma_1) \in \Gamma_f$. So, by saturation of rule (B_a), either $(\sigma_1\ \ \mathcal{M}'_1, u'_1; \ldots; \mathcal{M}'_i, u'_i\ \ \otimes) \in \Gamma_f$ or $((\sigma_1\ \ \mathcal{M}'_1, u'_1; \ldots; \mathcal{M}'_i, u'_i\ \ \checkmark) \in \Gamma_f$ and $(\sigma_1\ \ \mathcal{M}'_1, u'_1; \ldots; \mathcal{M}'_i, u'_i\ \ \psi) \in \Gamma_f)$. In the first case, by saturation of rule $(\otimes)$, either there is $k < i$ such that $((\sigma_1\ \mathcal{M}'_1, u'_1; \ldots; \mathcal{M}'_i, u'_i\ \neg Pre(w'_{k+1})) \in \Gamma_f$ and $(\sigma_1\ \mathcal{M}'_1, u'_1; \ldots; \mathcal{M}'_i, u'_i\ \checkmark) \in \Gamma_f)$ or $(\sigma_1\ \epsilon \otimes) \in \Gamma_f$. The case $(\sigma_1\ \mathcal{M}'_1, u'_1; \ldots; \mathcal{M}'_i, u'_i\ \neg Pre(w'_{k+1})) \in \Gamma_f$ and $(\sigma_1\ \mathcal{M}'_1, u'_1; \ldots; \mathcal{M}'_i, u'_i\ \checkmark) \in \Gamma_f)$ would entail by Induction Hypothesis that the sequence $(\mathcal{M}'_1, u'_1) \ldots, (\mathcal{M}'_k, u'_k)$ is executable in $(\mathcal{M}, w_{\sigma_1})$, but $\mathcal{M} \otimes \mathcal{M}'_1 \otimes \ldots \otimes \mathcal{M}'_k, (w_{\sigma_1}, u'_1, \ldots, u'_k) \models \neg Pre(w'_{k+1})$. Therefore, this would entail that $(w_{\sigma_1}, u'_1, \ldots, u'_k) \notin \mathcal{M} \otimes \mathcal{M}'_1 \otimes \ldots \otimes \mathcal{M}'_i$, which is also impossible. So, in both cases, we reach a contradiction. Hence, $(\sigma_1\ \ \mathcal{M}'_1, u'_1; \ldots; \mathcal{M}'_i, u'_i\ \ \otimes \notin \Gamma_f$ and therefore $(\sigma_1\ \ \mathcal{M}'_1, u'_1; \ldots; \mathcal{M}'_i, u'_i\ \ \checkmark) \in \Gamma_f$ and $(\sigma_1\ \mathcal{M}'_1, u'_1; \ldots; \mathcal{M}'_i, u'_i\ \psi) \in \Gamma_f$. Finally, because $\sum_{k=1}^{i} |\mathcal{M}'_k| + |\psi| < \sum_{k=1}^{i} |\mathcal{M}'_k| + |\varphi|$, we have by Induction Hypothesis that $\mathcal{M} \otimes \mathcal{M}'_1 \otimes \ldots \otimes \mathcal{M}'_i, (w_{\sigma_1}, u'_1, \ldots, u'_i) \models \psi$. We have proved this induction step.

- If $\varphi = [\mathcal{M}', w']\psi$, then by saturation of rule $([\mathcal{M}', w'])$, we have either $(\sigma\ \ \mathcal{M}'_1, w'_1; \ldots; \mathcal{M}'_i, w'_i; \mathcal{M}', w'\ \ \otimes) \in \Gamma_f$ or $((\sigma\ \ \mathcal{M}'_1, w'_1; \ldots; \mathcal{M}'_i, w'_i\ \ \neg[\mathcal{M}', w']\neg\psi) \in \Gamma_f$. In the first case, by saturation of the rule $(\otimes)$, either it holds that $(\sigma\ \ \mathcal{M}'_1, w'_1; \ldots; \mathcal{M}'_i, w'_i\ \ \otimes) \in \Gamma_f$ or it holds that $((\sigma\ \mathcal{M}'_1, w'_1; \ldots; \mathcal{M}'_i, w'_i\ \checkmark) \in \Gamma_f$ and $(\sigma\ \mathcal{M}'_1, w'_1; \ldots; \mathcal{M}'_i, w'_i\ \neg Pre(w')) \in \Gamma_f)$. The sub-case $(\sigma\ \mathcal{M}'_1, w'_1; \ldots; \mathcal{M}'_i, w'_i\ \otimes) \in \Gamma_f$ is impossible because by saturation of rule $(clash_{\checkmark,\otimes})$ and because $(\sigma\ \ \mathcal{M}'_1, w'_1; \ldots; \mathcal{M}'_i, w'_i\ \ \checkmark) \in \Gamma_f$, the branch carrying Γ_f would be closed. Therefore, the subcase where $(\sigma\ \mathcal{M}'_1, w'_1; \ldots; \mathcal{M}'_i, w'_i\ \checkmark) \in \Gamma_f$ and $(\sigma\ \mathcal{M}'_1, w'_1; \ldots; \mathcal{M}'_i, w'_i\ \neg Pre(w')) \in \Gamma_f$ holds. Now, because $\sum_{k=1}^{i} |\mathcal{M}'_k| + |\neg Pre(w')| < \sum_{k=1}^{i} |\mathcal{M}'_k| + |\varphi|$, by induction hypothesis, we have that the sequence $(\mathcal{M}'_1, w'_1), \ldots, (\mathcal{M}'_i, w'_i)$ is executable in $(\mathcal{M}, w_\sigma)$ and $\mathcal{M} \otimes \mathcal{M}'_1 \otimes \ldots \otimes \mathcal{M}'_i, (w_\sigma, w'_1, \ldots, w'_i) \models \neg Pre(w')$. Hence, $\mathcal{M} \otimes \mathcal{M}'_1 \otimes \ldots \otimes \mathcal{M}'_i, (w_\sigma, w'_1, \ldots, w'_i) \models [\mathcal{M}', w']\psi$. In the second case, we have $(\sigma\ \mathcal{M}'_1, w'_1; \ldots; \mathcal{M}'_i, w'_i\ \neg[\mathcal{M}', w']\neg\psi) \in \Gamma_f$. Then, by saturation of rule $(\neg[\mathcal{M}', w'])$, we have that $(\sigma\ \ \mathcal{M}'_1, w'_1; \ldots; \mathcal{M}'_i, w'_i; \mathcal{M}', w'\ \ \checkmark) \in \Gamma_f$ and $(\sigma\ \ \mathcal{M}'_1, w'_1; \ldots; \mathcal{M}'_i, w'_i; \mathcal{M}', w'\ \ \neg\neg\psi) \in \Gamma_f$. Then, by saturation of rule $(\neg\neg)$, we also have that $(\sigma\ \mathcal{M}'_1, w'_1; \ldots; \mathcal{M}'_i, w'_i; \mathcal{M}', w'\ \psi) \in \Gamma_f$. Now, $\sum_{k=1}^{i} |\mathcal{M}'_k| + |\mathcal{M}'| + |\psi| < \sum_{k=1}^{i} |\mathcal{M}'_k| + |\varphi| = \sum_{k=1}^{i} |\mathcal{M}'_k| + |\mathcal{M}'| + |\psi| + 1$. So, by Induction Hypothesis, we have that the sequence $(\mathcal{M}'_1, w'_1), \ldots, (\mathcal{M}'_i, w'_i), (\mathcal{M}', w')$ is executable in $(\mathcal{M}, w_\sigma)$ and $\mathcal{M} \otimes \mathcal{M}'_1 \otimes \ldots \otimes \mathcal{M}'_i \otimes \mathcal{M}', (w_\sigma, w'_1, \ldots, w'_i, w') \models \psi$.So, $\mathcal{M} \otimes \mathcal{M}'_1 \otimes \ldots \otimes \mathcal{M}'_i, (w_\sigma, w'_1, \ldots, w'_i) \models [\mathcal{M}', w']\psi$. Hence, in both cases, we have that the sequence $(\mathcal{M}'_1, w'_1), \ldots, (M'_i, w'_i)$ is executable in $(\mathcal{M}, w_\sigma)$ and $\mathcal{M} \otimes \mathcal{M}'_1 \otimes \ldots \otimes \mathcal{M}'_i, (w_\sigma, w'_1, \ldots, w'_i) \models [\mathcal{M}', w']\psi$.

- If $\varphi = \neg[\mathcal{M}', w']\psi$, then one can prove this induction step similarly to the second case of the previous induction step ($\varphi = [\mathcal{M}', w']\psi$).

- If $\varphi = [\pi \cup \gamma]\psi$, then by saturation of $([\pi \cup \gamma])$, we have that $(\sigma\ \ \mathcal{M}'_1, w'_1; \ldots; \mathcal{M}'_i, w'_i\ \ [\pi]\psi) \in \Gamma_f$ and also $(\sigma\ \mathcal{M}'_1, w'_1; \ldots; \mathcal{M}'_i, w'_i\ [\gamma]\psi) \in \Gamma_f$. Then, by Induction Hypothesis, the sequence $(\mathcal{M}'_1, w'_1), \ldots (\mathcal{M}'_i, w'_i)$ is executable in $(\mathcal{M}, w_\sigma)$, $\mathcal{M} \otimes \mathcal{M}'_1 \otimes \ldots \otimes \mathcal{M}'_i, (w_\sigma, w'_1, \ldots, w'_i) \models [\pi]\psi$ and $\mathcal{M} \otimes \mathcal{M}'_1 \otimes \ldots \otimes \mathcal{M}'_i, (w_\sigma, w'_1, \ldots, w'_i) \models [\gamma]\psi$. Therefore, $\mathcal{M} \otimes \mathcal{M}'_1 \otimes \ldots \otimes \mathcal{M}'_i, (w_\sigma, w'_1, \ldots, w'_i) \models [\pi \cup \gamma]\psi$.

- If $\varphi = \neg[\pi \cup \gamma]\psi$, then, by saturation of rule $(\neg[\pi \cup \gamma])$, either it holds that $(\sigma\ \ \mathcal{M}'_1, w'_1; \ldots; \mathcal{M}'_i, w'_i\ \ \neg[\pi]\psi) \in \Gamma_f$ or it holds that $(\sigma\ \ \mathcal{M}'_1, w'_1; \ldots; \mathcal{M}'_i, w'_i\ \ \neg[\gamma]\psi) \in \Gamma_f$. Without loss of generality, we can assume that $(\sigma\ \mathcal{M}'_1, w'_1; \ldots; \mathcal{M}'_i, w'_i\ \neg[\pi]\psi) \in \Gamma_f$. Then, by Induction Hypothesis, the sequence $(\mathcal{M}'_1, w'_1), \ldots, (\mathcal{M}'_i, w'_i)$ is executable in $(\mathcal{M}, w_\sigma)$ and $\mathcal{M} \otimes \mathcal{M}'_1 \otimes \ldots \otimes \mathcal{M}'_i, (w_\sigma, w'_1, \ldots, w'_i) \models \neg[\pi]\psi$. Hence, $\mathcal{M} \otimes \mathcal{M}'_1 \otimes \ldots \otimes \mathcal{M}'_i, (w_\sigma, w'_1, \ldots, w'_i) \models \neg[\pi \cup \gamma]\psi$. $\Box$

C. PROOF OF THEOREM 4

THEOREM 7. *The satisfiability problem of $\mathcal{L}_{DEL}$ is in NEXPTIME.*

PROOF. Let $\varphi \in \mathcal{L}_{DEL}$ and let $\mathcal{T}$ be the tableau tree for φ. The tableau method is a non-deterministic procedure. Indeed, rules $(\neg\wedge), (B_a), (\otimes), (\neg[\pi \cup \gamma])$ are non-deterministic rules. Each node of the tree $\mathcal{T}$ is the saturation of all rules concerning a specific symbol σ: $(\wedge), (\neg\neg), (\neg\wedge), (\bot), (\leftarrow_p), (\leftarrow_{\neg p}), (\neg[\mathcal{M}', w']), ([\mathcal{M}', w']), (\checkmark), (\otimes), ([\pi \cup \gamma]), (\neg[\pi \cup \gamma])$. There is an edge from a node corresponding to σ to a node corresponding to σ' if $(\sigma\ R_a \sigma')$ is in the set of terms in the current execution. In other terms, an edge corresponds to an application of the rule $(\neg B_a)$. Rule (B_a) is a propagation rule: from terms concerning σ, it adds terms concerning σ' where $(\sigma\ R_a \sigma')$. In this proof, we show that:

1. at each node of $\mathcal{T}$ concerning a specific symbol σ, there are at most an exponential number of terms;
2. the arity of $\mathcal{T}$ is at most exponential;
3. the depth of $\mathcal{T}$ is linear.

Each step of the algorithm is an application of a rule and it will add new terms. So, the construction of the tree $\mathcal{T}$ is done non-deterministically in exponential time.

$\boxed{\text{Proof of 1}}$ Except terms of the form $(\sigma\ R_a\ \sigma')$, all other terms are of the form $(\sigma\ \Sigma'\ \psi)$ where:

- ψ is a formula in a closure $CL(\varphi)$ of the initial formula φ (this includes the fact that ψ can be in the closure of a precondition appearing in an event model of φ);
- Σ' is a list of pointed event models $\mathcal{M}'_1, w'_1, \ldots, \mathcal{M}'_i, w'_i$.

Formally, the closure $CL(\varphi)$ of a formula φ is the smallest set of formulas containing φ and satisfying all those constraints:

- if $\neg\psi \in CL(\varphi)$, then $\psi \in CL(\varphi)$;
- if $\chi \wedge \psi \in CL(\varphi)$, then $\chi, \psi, \neg\chi, \neg\psi \in CL(\varphi)$;
- if $B_a\psi \in CL(\varphi)$, then $\neg\psi, \psi \in CL(\varphi)$;
- if $[\mathcal{M}', w']\psi \in CL(\varphi)$, then $\psi, \neg\psi \in CL(\varphi)$ and all the precondition formulas are in $CL(\varphi)$, that is, if $\mathcal{M}' = (W', R', Pre)$, then for all $w' \in W'$, $Pre(w') \in CL(\varphi)$;
- if $[\pi \cup \gamma]\psi \in CL(\varphi)$ then $[\pi]\psi, [\gamma]\psi \in CL(\varphi)$.

The cardinal of the closure $CL(\varphi)$ is linear in $|\varphi|$.

Now let us give an upper bound for the cardinal of the set of possible lists Σ' of pointed event models. Such a list is of the following form $\mathcal{M}'_1, u'_1, \ldots, \mathcal{M}'_n, u'_n$, where:

- n is bounded by the size of φ;
- $\mathcal{M}'_i$ are event models that appear in the formula φ;
- each u'_i are possible events of $\mathcal{M}'_i$.

Let us assume that n is chosen. The number of event models appearing in φ is bounded by $|\varphi|$, so there are at most $|\varphi|$ choices for $\mathcal{M}'_1$. The number of possible choices for u'_1 is bounded by $|\mathcal{M}'_1| \leq |\varphi|$. So there are at most $|\varphi|$ choices for u'_1. Hence, there are at most $|\varphi|^2$ choices for $\mathcal{M}'_1, u'_1$. We can repeat the same reasoning for counting the number of choices for $\mathcal{M}'_2, u'_2, \ldots, \mathcal{M}'_{n-1}, u'_{n-1}$ and $\mathcal{M}'_n, u'_n$. By multiplying all the number of choices, we see that there are at most $(|\varphi|^2)^n = O(|\varphi|^{2|\varphi|})$ possible lists Σ' for a given n. Since there are at most $|\varphi|$ choices for n, there are at most $|\varphi| \times O\left(|\varphi|^{2|\varphi|}\right) = O\left(|\varphi|^{2|\varphi|+1}\right) = O\left(2^{|\varphi|^2}\right)$ possible lists Σ'. So, the number of possible sequences Σ' is exponential in the size of φ.

Combining these two intermediary results, we obtain that the set of all possible terms in a given node of $\mathcal{T}$ is in $O(|\varphi|) \times O\left(2^{|\varphi|^2}\right) = O\left(2^{|\varphi|^2}\right)$, so at most exponential in the size of φ.

$\boxed{\text{Proof of 2}}$ The worse case is when there is an exponential number of terms of the form $(\sigma\ \Sigma'\ \neg B_a\psi)$. Indeed, in that case, an exponential number of new symbols σ_{new} will arise from the applications of rule $(\neg B_a)$ on terms concerning σ. In other words, the arity of $\mathcal{T}$ is at most exponential in the size of φ.

$\boxed{\text{Proof of 3}}$ First, we warn the reader about the following fact: it is false that the modal depth of the subformulas decreases strictly from a node to its children, because of the rules $(\checkmark)$ and $(\otimes)$. The idea is that once all modal operators of the form $B_a\psi$ and $[\mathcal{M}', w']\psi$ have been treated, the rules $(\checkmark)$ and $(\otimes)$ stop to be applicable for the corresponding formulas.

Let us define a quantity that is strictly decreasing during the execution of the tableau method. Let us consider a given term $(\sigma\ \Sigma'\ \psi)$. The size of a term $t = (\sigma\ \Sigma'\ \psi)$ is defined by

$$|t| = 1 + \sum_{(\mathcal{M}', w') \in \Sigma'} \left(|\mathcal{M}'| + 1\right) + |\psi|$$

where $|\checkmark| = |\otimes| = 1$ and $(\mathcal{M}', w') \in \Sigma'$ means that the event model $(\mathcal{M}', w')$ appears in Σ'.

Now, following Fitting's terminology [Fitting, 1983], all the rules of the tableau method are *strictly analytic*: all terms t that are in the denominator of a rule are such that there exists a term c in the numerator such that $|t| < |c|$. For instance let us consider the following rule:

$$\frac{(\sigma\ \Sigma'\ [\mathcal{M}', w']\varphi)}{(\sigma\ \Sigma'\ ; \mathcal{M}', w'\ \ \otimes) \;\Big|\; \begin{array}{l}(\sigma\ \Sigma'\ ; \mathcal{M}', w'\ \ \checkmark)\\(\sigma\ \Sigma'\ ; \mathcal{M}', w'\ \ \neg\varphi)\end{array}}\ ([\mathcal{M}', w'])$$

We have $|(\sigma\ \Sigma'\ ; \mathcal{M}', w'\ \ \otimes)| < A$, $|(\sigma\ \Sigma'\ ; \mathcal{M}', w'\ \ \checkmark)| < A$ and $|(\sigma\ \Sigma'\ ; \mathcal{M}', w'\ \ \neg\varphi)| < A$ where $A = |(\sigma\ \Sigma'\ [\mathcal{M}', w']\varphi)|$.

To each node of the tree $\mathcal{T}$ that corresponds to a given symbol σ, we attach the following quantity: the maximum of $|t|$ where t is a term concerning σ. As rules are strictly analytic, this quantity will decrease strictly on a branch of the tree. Furthermore, for the root of the tree, this quantity is $|(\sigma_0\ \epsilon\ \varphi)| = 1 + |\varphi|$. As this quantity is positive, it proves the terminaison of the procedure and also that the depth of the tree $\mathcal{T}$ is linear in the size of φ.

To sum up, the depth of the tree $\mathcal{T}$ is linear whereas the arity may be exponential. As a consequence, the tree $\mathcal{T}$ has at most an exponential number of nodes in the size of φ. Indeed, because of point 1, the number of nodes of the tree $\mathcal{T}$ is in $O\left(\left(2^{|\varphi|^2}\right)^{|\varphi|}\right) = O\left(2^{|\varphi|^3}\right)$ which is exponential in the size of φ. Moreover, constructing such a tree $\mathcal{T}$ can be done non-deterministically in an exponential amount of time, for each execution of the tableau method. So, the procedure is in NEXPTIME. $\square$