Complexity of Dynamic Epistemic Logic with Common Knowledge

Tristan Charrier

Univ Rennes
IRISA
France

François Schwarzentruber

Univ Rennes
IRISA
France

Abstract

We consider the language of Dynamic epistemic logic with knowledge operators, common knowledge operators and dynamic operators based on event models. First, we prove that the model checking remains PSPACE-complete when common knowledge is added. Second, we prove that the satisfiability problem is 2EXPTIME-complete. We further address the model checking and the satisfiability problem for succinct inputs: we prove that complexities remain unchanged.

Keywords: Dynamic epistemic logic, common knowledge, complexity theory.

1 Introduction

Dynamic epistemic logic (DEL) [29] is a framework for reasoning about knowledge and complex actions (public announcement, public actions, private announcements, etc.). On top of that, *common knowledge* (everybody knows that everybody knows that...) is a condition for agents to act simultaneously in distributed systems [14] , in games [2] and more generally in artificial intelligence and computer science ([20], p. 45).

In this paper, we tackle both the model checking problem and the satisfiability problem of DEL with common knowledge. Both notions are relevant and have their advantages and drawbacks (see [16]).

- *Model checking.* It consists in checking whether a specification is true in a specific multi-agent system, described here by means of a Kripke model. Model checking allows for solving epistemic planning [5] but in its bounded version. For instance, one may check that formula $\varphi_{plan?}$ defined by $\langle\{e_1, e_2\}\rangle^n C_G p$ (there is a plan of length n made up of actions e_1 or

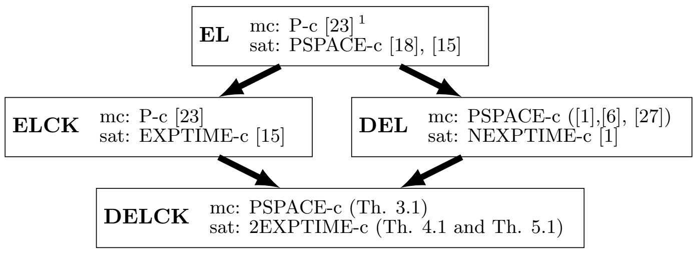

Figure 1. Complexities for the model checking (mc) and the satisfiability (sat) problem.

e_2 that leads to the common knowledge of p). Model checking has low complexity in general but we only reason about one fully-described state.

- *Satisfiability problem.* It consists in checking consistency of a specification. It allows for solving bounded planning/games not in only one initial state but in a *class* of initial states, described by a formula φ_{init}. Typically, we check that $\varphi_{init} \rightarrow \varphi_{plan?}$ is valid (by proving that the negation of it is unsatisfiable). Unfortunately, compared to model checking, the satisfiability problem is more computationally expensive in general.

The use of complex formulas is relevant for checking bounded games instead of bounded plans, by alternating diamond and box dynamic operators ($\langle\{e_1, e_2\}\rangle[\{e_3, e_4\}]C_p$). Furthermore, we can handle bounded implicit coordinated plans [12], where each agent a_i executing an event e_i knows that the rest of the plan will be correct, in the following sense: a formula of the form $K_{a_1}\langle e_1\rangle K_{a_2}\langle e_2\rangle \dots \varphi_G$ where φ_G is a goal formula, is true.

The exact complexities were first investigated in [1] but the problem was left open for common knowledge. Other papers address the model checking problem but always without the common knowledge operator ([6], [27]). Figure 1 shows the complexities of Dynamic epistemic logics (**EL**: epistemic logic without common knowledge, **ELCK**: epistemic logic with common knowledge, **DEL**: dynamic epistemic logic without common knowledge, **DELCK**: dynamic epistemic logic with common knowledge). Our paper solves open problems for the complexity of **DELCK**:

- We show that model checking remains in PSPACE when formulas contain common knowledge operators. The algorithm follows the same principle than the one given in [1]. Interestingly, common knowledge is treated via the *divide-and-conquer* paradigm [13]. Compared to [1], we also treat postconditions in event models. The contribution is a neat analysis of the complexity when event model update and common knowledge are mixed.
- We provide a double-exponential algorithm for the satisfiability problem of **DELCK** by using the method Pratt developed for Propositional Dynamic

[1] Hardness was proven for the EX, AX-fragment of temporal logic CTL, which is technically the same logic as epistemic logic K.

Logic [22].

- We show that the satisfiability problem with common-knowledge is actually 2EXPTIME-hard, already for public actions, and no constraint on the frames, by providing a reduction from the halting problem of an alternating Turing machine running in exponential space (see [7]).
- Succinctness is a key concept in complexity theory (see [21]) and is central in symbolic model checking. As shown in [10], a succinct representation for event models is also relevant for modeling complex actions such as attention-based announcements. Therefore, succinctness is even relevant for the satisfiability problem where formulas contain event models presented in a succinct way. We show that our complexity results for model checking and for the satisfiability problem still hold for succinct representations of Kripke models and event models. Incidentally, we simplify the formalism introduced in [10].

The results are proven for arbitrary models but they also hold for **S5** models, in which epistemic relations are equivalence relations. In Section 2, we define the logic **DELCK** and alternating Turing machines, which are relevant for the proofs on the other sections. In Section 3 we prove that the model checking problem against **DELCK** is PSPACE-complete. In Sections 4 and 5 we prove that the SAT problem of **DELCK** is in 2-EXPTIME and is 2-EXPTIME-hard respectively. In Section 6 we prove that the same complexity results hold when the input models are represented succinctly. Finally, in Section 7 we discuss related work and conclude.

2 Background

2.1 Background on dynamic epistemic logic

We consider a countable set of atomic propositions AP and a finite set of agents Ag. We recall that the language of epistemic logic with common knowledge (**ELCK**), denoted by $\mathcal{L}_{\mathbf{ELCK}}$, is defined by the following BNF:

$$\varphi ::= \top \mid p \mid \neg\varphi \mid (\varphi \vee \varphi) \mid K_a\varphi \mid C_G\varphi$$

with $p \in AP$, $a \in Ag$, $G \subseteq Ag$. Formula $K_a\varphi$ reads as "agent a knows that φ holds" and $C_G\varphi$ reads as "φ is common knowledge among the agents of G". As usual, we define the abbreviations $(\varphi_1 \wedge \varphi_2)$ for $\neg(\neg\varphi_1 \vee \neg\varphi_2)$, $\hat{K}_a\varphi$ for $\neg K_a\neg\varphi$ and $\hat{C}_G\varphi$ for $\neg C_G\neg\varphi$.

Definition 2.1 A *Kripke model* $\mathcal{M} = (W, (R_a)_{a\in Ag}, V)$ is defined by a non-empty set W of epistemic worlds, epistemic relations $(R_a)_{a\in Ag} \subseteq W \times W$ and a valuation function $V : W \to 2^{AP}$.

We write $R_a^{\mathcal{M}}$ for the epistemic relation for agent a in model $\mathcal{M}$. A pair $(\mathcal{M}, w)$ is called a *pointed epistemic model.* The left part of Figure 2 shows a pointed epistemic model $\mathcal{M}, w$ with two words w and u. As usual, formulas are interpreted in pointed epistemic models and we define $\mathcal{M}, w \models \varphi$ (φ is true in $\mathcal{M}, w$) by induction on φ (Boolean cases are omitted):

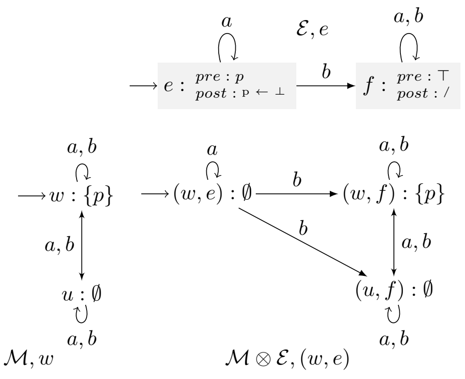

Figure 2. Example of product.

- $\mathcal{M}, w \models p$ if $p \in V(w)$;
- $\mathcal{M}, w \models K_a\varphi$ if for all $u \in W, wR_au$ implies $\mathcal{M}, u \models \varphi$;
- $\mathcal{M}, w \models C_G\varphi$ if for all $u \in W, wR_Gu$ implies $\mathcal{M}, u \models \varphi$ where R_G is the transitive and reflexive closure of $\bigcup_{a\in G} R_a$.

As usual, we define $\varphi_1 \wedge \varphi_2 = \neg(\neg\varphi_1 \vee \neg\varphi_2)$, $\varphi_1 \rightarrow \varphi_2 = \neg\varphi_1 \vee \varphi_2$ and $\varphi_1 \leftrightarrow \varphi_2 = (\varphi_1 \rightarrow \varphi_2) \wedge (\varphi_2 \rightarrow \varphi_1)$. The dynamic of the system is modeled by event models. An event model is like a Kripke model but epistemic worlds are replaced by events labeled by a precondition and a postcondition.

Definition 2.2 An *event model* $\mathcal{E} = (\mathsf{E}, (R_a^{\mathcal{E}})_{a\in Ag}, pre, post)$ is defined by a non-empty set of *events* E, epistemic relations $(R_a^{\mathcal{E}})_{a\in Ag} \subseteq \mathsf{E}\times\mathsf{E}$, a precondition function $pre : \mathsf{E} \rightarrow \mathcal{L}_{\mathbf{ELCK}}$ and a postcondition function $post : \mathsf{E} \times AP \rightarrow \mathcal{L}_{\mathbf{ELCK}}$.

A pair $(\mathcal{E}, e)$ with $e \in \mathcal{E}$ is called a *pointed event model*, where e represents the actual event. A pair $(\mathcal{E}, \mathsf{E}_0)$ with $\mathsf{E}_0 \subseteq \mathsf{E}$ is called a *multi-pointed event model*, where E_0 represents the set of possible actual events. Pointed event models correspond to deterministic actions and multi-pointed event models correspond non-deterministic actions. We may confuse $(\mathcal{E}, e)$ and $(\mathcal{E}, \{e\})$.

Example 2.3 The top of Figure 2 shows an example of a pointed event model with two events e and f. The actual event is e but agent b imagines event f as the sole possible event.

Definition 2.4 Let $\mathcal{M} = (W, (R_a)_{a\in Ag}, V)$ be a Kripke model. Let $\mathcal{E} = (\mathsf{E}, (R_a^{\mathcal{E}})_{a\in Ag}, pre, post)$ be an event model. The *product* of $\mathcal{M}$ and $\mathcal{E}$ is $\mathcal{M} \otimes \mathcal{E} = (W', (R_a)', V')$ where:

- $W' = \{(w, e) \in W \times \mathsf{E} \mid \mathcal{M}, w \models pre(e)\}$;
- $(w, e)R'_a(w', e')$ iff wR_aw' and $eR_a^{\mathcal{E}}e'$;
- $V'((w, e)) = \{p \in AP \mid \mathcal{M}, w \models post(e, p)\}$.

Example 2.5 Figure 2 shows the product operation.

An event is e is said *executable* in a world w if its precondition $pre(e)$ holds in w. The language $\mathcal{L}_{\mathbf{DELCK}}$ extends $\mathcal{L}_{\mathbf{ELCK}}$ with dynamic modalities and is defined by the following BNF:

$$\varphi ::= \top \mid p \mid \neg\varphi \mid (\varphi \vee \varphi) \mid K_a\varphi \mid C_G\varphi \mid \langle \mathcal{E}, \mathsf{E}_0 \rangle \varphi$$

with $p \in AP$, $a \in Ag$. Formula $\langle \mathcal{E}, \mathsf{E}_0 \rangle \varphi$ reads as "There is an executable event in E_0 and φ holds after having executed it". In [29], the event models can feature any formula of $\mathcal{L}_{\mathbf{DELCK}}$, not just $\mathcal{L}_{\mathbf{ELCK}}$. The results of the paper still hold for this definition, but for the sake of simplicity, we consider event models to not feature dynamic constructions in preconditions and postconditions. We define the dual construction $[\mathcal{E}, \mathsf{E}_0]\varphi$ for $\neg\langle \mathcal{E}, \mathsf{E}_0 \rangle \neg\varphi$, that is read as "For all executable events in E_0, φ holds after having executed it".

Definition 2.6 We extend the definition $\mathcal{M}, w \models \varphi$ to $\mathcal{L}_{\mathbf{DELCK}}$ with:

- $\mathcal{M}, w \models \langle \mathcal{E}, \mathsf{E}_0 \rangle \varphi$ if there exists $e \in \mathsf{E}_0$ s.t. $\mathcal{M}, w \models pre(e)$ and $\mathcal{M} \otimes \mathcal{E}, (w, e) \models \varphi$.

A formula φ is satisfiable iff there exists a pointed epistemic model $\mathcal{M}, w$ such that $\mathcal{M}, w \models \varphi$. In the sequel, we suppose w.l.o.g. that, given a formula φ, there is a common $\mathcal{E}$ and dynamic operators $\langle \mathcal{E}, \mathsf{E}_0 \rangle$ and $[\mathcal{E}, \mathsf{E}_0]$ are written $\langle \mathsf{E}_0 \rangle$ and $[\mathsf{E}_0]$ respectively. The idea is to set $\mathcal{E}$ to be the disjoint union of all occurrences of event models in the formula. E.g. formula $\langle \mathcal{E}_1, e_1 \rangle K_a p \wedge \langle \mathcal{E}_2, e_2 \rangle K_a q$ is rewritten as $\langle e_1 \rangle K_a p \wedge \langle e_2 \rangle K_a q$ with a common event model $\mathcal{E}$ defined as the disjoint union of $\mathcal{E}_1$ and $\mathcal{E}_2$.

In the sequence, we take the abbreviation $w\overrightarrow{e}$ for $(w, e_1, \ldots, e_n)$ and $\mathcal{M}\overrightarrow{\mathcal{E}}$ for $\mathcal{M}, \mathcal{E}_1, \ldots, \mathcal{E}_n$. We write $\mathcal{M}\overrightarrow{\mathcal{E}}, w\overrightarrow{e} \models \varphi$ instead of $\mathcal{M} \otimes \mathcal{E}_1 \otimes \cdots \otimes \mathcal{E}_n, (\ldots(w, e_1), \ldots, e_n) \models \varphi$. A sequence of events $\overrightarrow{e}$ is executable in w if $w\overrightarrow{e}$ is in $\mathcal{M}\overrightarrow{\mathcal{E}}$. The empty sequence of events is denoted by ϵ. The empty sequence is of course executable in all worlds.

2.2 Background on alternation

In Sections 3 and 5, we will make use of *alternation* [8]. Formally, an *alternating Turing machine* is a tuple $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{acc}, q_{rej}, g)$ where:

Q is the finite set of states of M; Σ is the finite input alphabet; Γ is the finite tape alphabet with $\Sigma \subseteq \Gamma$; $\delta \subseteq Q \times \Gamma \times Q \times \Gamma \times \{-1, +1\}$ is the transition function; $q_0 \in Q$ is the initial state, $q_{acc} \in Q$ the accepting state and $q_{rej} \in Q$ is the rejecting state; $g : Q \to \{\exists, \forall\}$ is the quantification function for the states.

We suppose that only q_{acc} and q_{rej} are end states (i.e. have no outgoing transitions to other states). The execution of the machine is controlled by two players $\exists$ and $\forall$. When the current state q is existential ($g(q) = \exists$) (resp. universal ($g(q) = \forall$)), player $\exists$ (resp. $\forall$) chooses the transition to apply. A configuration is accepting if player $\exists$ has a winning strategy for reaching the accepting state. An input word is accepted by the machine if the corresponding initial configuration is *accepting*. Chandra and Stockmeyer [8] defined complexity classes with respect to alternating Turing machines. E.g. APTIME is the class of decision problems decided by an alternating Turing machine in polynomial time (the height of the computation tree is polynomial in the size of the input). They proved that APTIME = PSPACE and AEXPSPACE = 2-EXPTIME [8].

The original definition of alternating Turing machine [8] also allows for *negative states.* When the current state q is negative, the acceptance condition is negated. We can get rid off negative states without changing the definition of complexity classes (see [17], Lecture 7, Lemma 7.3, p. 47).

3 Model checking

The model checking for **DELCK**, given a pointed Kripke model $\mathcal{M}, w$, a formula Φ of $\mathcal{L}_{\mathbf{DELCK}}$, asks to decide whether $\mathcal{M}, w \models \Phi$. In this section, we prove the following theorem.

Theorem 3.1 *The model checking problem for* **DELCK** *is* PSPACE*-complete.*

Hardness comes directly from the PSPACE-hardness of the model checking of **DEL** without common knowledge [1] (actually, it is already PSPACE-hard for single-pointed event models [6], but actually even when the Kripke model is **S5** and event models are **S5** and single-pointed [27]). For the PSPACE-membership, Figure 3 provides the pseudo-code of an alternating Turing machine that decides the model checking problem for **DELCK** in polynomial time. In the pseudo-code in Figure 3, existential (∃) and keyword **or** (resp. universal (∀) choices and keyword **and**) corresponds to existential (resp. universal) states. Keyword **not** corresponds to a negated state. The upper bound is proven since PSPACE = APTIME. The machine starts by calling $mc(\mathcal{M}, w, \Phi)$. The specifications of the procedures mc, $inval$ in, rel and rel^* (see Figure 3) are given in the following proposition:

Proposition 3.2 *For all formulas φ, for all Kripke models $\mathcal{M}$, for all sequences of event models $\overrightarrow{\mathcal{E}}$, for all worlds $w\overrightarrow{e}, u\overrightarrow{f}$ of $\mathcal{M}\overrightarrow{\mathcal{E}}$, for all agents a, for all groups of agents G, for integers i that are powers of two,*

$mc(\mathcal{M}\overrightarrow{\mathcal{E}}, w\overrightarrow{e}, \varphi)$ *is accepting*	*iff* $\mathcal{M}\overrightarrow{\mathcal{E}}, w\overrightarrow{e} \models \varphi$,
$inval(p, w\overrightarrow{e}, \mathcal{M}\overrightarrow{\mathcal{E}})$ *is accepting*	*iff* $p \in V(w\overrightarrow{e})$,
$in(w\overrightarrow{e}, \mathcal{M}\overrightarrow{\mathcal{E}})$ *is accepting*	*iff* $w\overrightarrow{e} \in \mathcal{M}\overrightarrow{\mathcal{E}}$,
$rel(w\overrightarrow{e}, u\overrightarrow{f}, a, \mathcal{M}\overrightarrow{\mathcal{E}})$ *is accepting*	*iff* $(w\overrightarrow{e}, u\overrightarrow{f}) \in R_a$,
and $rel^*(w\overrightarrow{e}, u\overrightarrow{f}, G, i, \mathcal{M}\overrightarrow{\mathcal{E}})$ *is accepting*	*iff* $(w\overrightarrow{e}, u\overrightarrow{f}) \in \bigcup_{j \leq i} \left(\bigcup_{a \in G} R_a\right)^j$.

Proof The proposition is straightforwardly proven by induction since the pseudo-code directly reflects the semantics of **DELCK**. The only difficulties are:

- The induction works on the quantities given in Figure 3 and thanks to the following Lemma 3.4.
- $\bigcup_{j \leq B_{\mathcal{M},\Phi}} \left(\bigcup_{a \in G} R_a\right)^j = \left(\bigcup_{a \in G} R_a\right)^*$ since the number of worlds in $\mathcal{M}\overrightarrow{\mathcal{E}}$ in bounded by $B_{\mathcal{M},\Phi}$;
- The design of Procedure rel^* relies on the *divide and conquer* paradigm. For checking that $u\overrightarrow{f}$ is reachable by at most i $\bigcup_{a \in G} R_a$-steps from $w\overrightarrow{e}$,

```
procedure $mc(\mathcal{M}\overrightarrow{\mathcal{E}}, w\overrightarrow{e}, \varphi)$    $|\mathcal{M}\overrightarrow{\mathcal{E}}| + |\varphi|$
    case $\varphi = p$: $inval(p, \mathcal{M}\overrightarrow{\mathcal{E}}, w\overrightarrow{e})$
    case $\varphi = (\varphi_1 \vee \varphi_2)$: (∃) choose $i \in \{1,2\}$; $mc(\mathcal{M}\overrightarrow{\mathcal{E}}, w\overrightarrow{e}, \varphi_i)$
    case $\varphi = \neg\psi$: not $mc(\mathcal{M}\overrightarrow{\mathcal{E}}, w\overrightarrow{e}, \psi)$.
    case $\varphi = K_a\psi$:
        (∀) choose $u\overrightarrow{f} \in \mathcal{M}\overrightarrow{\mathcal{E}}$
        (∃) not $in(u\overrightarrow{f}, \mathcal{M}\overrightarrow{\mathcal{E}})$ or not $rel(w\overrightarrow{e}, u\overrightarrow{f}, a, \mathcal{M}\overrightarrow{\mathcal{E}})$
            or $mc(\mathcal{M}\overrightarrow{\mathcal{E}}, u\overrightarrow{f}, \psi)$
    case $\varphi = \langle \mathsf{E}_0 \rangle \psi$:
        (∃) choose $e \in \mathsf{E}_0$; (∀) $mc(\mathcal{M}\overrightarrow{\mathcal{E}}, w\overrightarrow{e}, pre(e))$
            and $mc(\mathcal{M}\overrightarrow{\mathcal{E}} :: \mathcal{E}, w\overrightarrow{e} :: e, \psi)$.
    Case $\varphi = C_G\psi$:
        (∀) choose $u\overrightarrow{f} \in \mathcal{M}\overrightarrow{\mathcal{E}}$
        (∃) not $in(u\overrightarrow{f}, \mathcal{M}\overrightarrow{\mathcal{E}})$ or not $rel^*(w\overrightarrow{e}, u\overrightarrow{f}, G, B_{\mathcal{M},\Phi}, \mathcal{M}\overrightarrow{\mathcal{E}})$
            or $mc(\mathcal{M}\overrightarrow{\mathcal{E}}, u\overrightarrow{f}, \psi)$
```

```
procedure $inval(p, w\overrightarrow{e}, \mathcal{M}\overrightarrow{\mathcal{E}})$    $|\mathcal{M}\overrightarrow{\mathcal{E}}|$
    case $\overrightarrow{\mathcal{E}} = \epsilon$: if $p \in V(w)$ then accept else reject
    case $\overrightarrow{\mathcal{E}} = \overrightarrow{\mathcal{E}}'::\mathcal{E}$ and $w\overrightarrow{e} = w\overrightarrow{e}'::e$: $mc(\mathcal{M}\overrightarrow{\mathcal{E}}', w\overrightarrow{e}', post(e,p))$
```

```
Procedure $in(w\overrightarrow{e}, \mathcal{M}\overrightarrow{\mathcal{E}})$    $|\mathcal{M}\overrightarrow{\mathcal{E}}|$
    case $\overrightarrow{\mathcal{E}} = \epsilon$: accept
    case $\overrightarrow{\mathcal{E}} = \overrightarrow{\mathcal{E}}'::\mathcal{E}$ and $w\overrightarrow{e} = w\overrightarrow{e}'::e$: (∀) $mc(\mathcal{M}\overrightarrow{\mathcal{E}}', w\overrightarrow{e}', pre(e))$
                                                   and $in(w\overrightarrow{e}', \mathcal{M}\overrightarrow{\mathcal{E}}')$
```

```
procedure $rel(w\overrightarrow{e}, u\overrightarrow{f}, a, \mathcal{M}\overrightarrow{\mathcal{E}})$    $|\mathcal{M}\overrightarrow{\mathcal{E}}|$
    case $\overrightarrow{\mathcal{E}} = \epsilon$: if $(w,u) \in R_a^{\mathcal{M}}$ then accept else reject
    case $\overrightarrow{\mathcal{E}} = \overrightarrow{\mathcal{E}}' :: \mathcal{E}$, $\overrightarrow{e} = \overrightarrow{e}' :: e$ and $\overrightarrow{f} = \overrightarrow{f}' :: f$:
        (∀) $rel(w\overrightarrow{e}', u\overrightarrow{f}', a, \mathcal{M}\overrightarrow{\mathcal{E}}')$ and if $(e,f) \in R_a^{\mathcal{E}}$ then accept
                                                   else reject
```

```
procedure $rel^*(w\overrightarrow{e}, u\overrightarrow{f}, G, i, \mathcal{M}\overrightarrow{\mathcal{E}})$    $|\mathcal{M}\overrightarrow{\mathcal{E}}| + \log i$
    case $i = 1$: if $u\overrightarrow{f} = w\overrightarrow{e}$ then accept else (∃) choose $a \in G$;
                                                   $rel(w\overrightarrow{e}, u\overrightarrow{f}, a, \mathcal{M}\overrightarrow{\mathcal{E}})$
    case $i \geq 2$:
        (∃) choose $v\overrightarrow{g} \in \mathcal{M}\overrightarrow{\mathcal{E}}$
        (∀) $in(v\overrightarrow{g}, \mathcal{M}\overrightarrow{\mathcal{E}})$ and $rel^*(w\overrightarrow{e}, v\overrightarrow{g}, G, i/2, \mathcal{M}\overrightarrow{\mathcal{E}})$ and
        $rel^*(v\overrightarrow{g}, u\overrightarrow{f}, G, i/2, \mathcal{M}\overrightarrow{\mathcal{E}})$
```

Figure 3. Model checking procedures for **DELCK** (in gray: quantities associated to each procedure call).

we guess an intermediate world $v\overrightarrow{g} \in \mathcal{M}\overrightarrow{\mathcal{E}}$ and check that $v\overrightarrow{g}$ is reachable by at most $\frac{i}{2} \bigcup_{a\in G} R_a$-steps from $w\overrightarrow{e}$ and that $u\overrightarrow{f}$ is reachable by at most $i/2 \bigcup_{a\in G} R_a$-steps from $v\overrightarrow{g}$.

□

Better than giving a tedious and straightforward proof of Proposition 3.2, let us explain the algorithm on the example of $mc(\mathcal{M}, w, \langle\mathcal{E}, \mathsf{E}_0\rangle\neg C_G p)$.The procedure starts by choosing e in E_0. Then, we check that both $pre(e)$ holds in w and that $\neg C_G p$ holds in we. Checking that $\neg C_G p$ holds in we leads to a negated configuration: we negate the fact that $C_G p$ holds in we. It is followed by a universal choice of $u\overrightarrow{f} \in \mathcal{M}\overrightarrow{\mathcal{E}}$. For each choice of $uf \in \mathcal{ME}$, we progress in an existential configuration that checks that either uf is not a world of $\mathcal{ME}$, either wf is not reachable from we by at most $B_{\mathcal{M},\Phi} \bigcup_{a\in G} R_a^{\mathcal{ME}}$-steps or that p in uf. Checking that p holds in uf is performed by the call of $inval(p, uf, \mathcal{ME})$, which itself check that the postcondition $post(e, p)$ holds in u.

The quantities associated to each procedure call used for the proofs by induction require a careful definition. They depend on the input $(\mathcal{M}, w, \Phi)$ of the model checking problem. Let $B_{\mathcal{M},\Phi}$ be the smallest power of two that is greater than the number of worlds in Kripke model $\mathcal{M}\overrightarrow{\mathcal{E}}$ where $\overrightarrow{\mathcal{E}}$ is the list of all event models appearing in the formula Φ.

Definition 3.3 $|\mathcal{M}|$, $|\mathcal{E}|$ and $|\varphi|$ are defined by mutual induction. First, $|\mathcal{M}|$ (resp. $|\mathcal{E}|$) are the number of bits to encode Kripke model $\mathcal{M}$ (resp. event model $\mathcal{E}$). In particular, $|\mathcal{E}|$ takes into account the memory needed to store the precondition and the postcondition functions. Then, $|\mathcal{M}\overrightarrow{\mathcal{E}}|$ denotes $|\mathcal{M}| + \sum_{i=1}^{n} |\mathcal{E}_i|$. Second $|\varphi|$ denotes the length of φ, defined by induction as usual except for the two following cases:

- $|\langle\mathcal{E}, \mathsf{E}_0\rangle\varphi| := |\mathcal{E}| + 1 + |\varphi|$;
- $|C_G\varphi| := \log_2 B_{\mathcal{M},\Phi} + 1 + |\varphi|$.

Lemma 3.4 *The quantities given in gray in Figure 3 are strictly decreasing along a branch of the computation tree.*

Proof Let us discuss the following cases (the other ones are left to the reader):

- The quantity for $mc(\mathcal{M}\overrightarrow{\mathcal{E}}, w\overrightarrow{e}, C_G\varphi)$ is $|\mathcal{M}\overrightarrow{\mathcal{E}}| + |C_G\varphi| + 1 = |\mathcal{M}\overrightarrow{\mathcal{E}}| + \log_2 B_{\mathcal{M},\Phi} + |\varphi| + 1$ and is strictly greater than the quantity for $rel^*(w\overrightarrow{e}, u\overrightarrow{f}, G, B_{\mathcal{M},\Phi}, \mathcal{M}\overrightarrow{\mathcal{E}})$, which is $|\mathcal{M}\overrightarrow{\mathcal{E}}| + \log_2 B_{\mathcal{M},\Phi}$.
- The quantity for $mc(\mathcal{M}\overrightarrow{\mathcal{E}}, w\overrightarrow{e}, \langle\mathcal{E}, \mathsf{E}_0\rangle\varphi)$ is $|\mathcal{M}\overrightarrow{\mathcal{E}}| + |\mathcal{E}| + \varphi| + 1$ and is strictly greater than the quantity for $mc(\mathcal{M}\overrightarrow{\mathcal{E}}, w\overrightarrow{e}, pre(e))$, which is $|\mathcal{M}\overrightarrow{\mathcal{E}}| + |pre(e)| < |\mathcal{M}\overrightarrow{\mathcal{E}}| + |\mathcal{E}|$.
- The quantity for $inval(p, w\overrightarrow{e}, \mathcal{M}\overrightarrow{\mathcal{E}})$ is $|\mathcal{M}\overrightarrow{\mathcal{E}}'\mathcal{E}| = |\mathcal{M}\overrightarrow{\mathcal{E}}'| + |\mathcal{E}|$ and is strictly greater than the quantity for $mc(\mathcal{M}\overrightarrow{\mathcal{E}}', w\overrightarrow{e}', post(e, p))$ which $|\mathcal{M}\overrightarrow{\mathcal{E}}'| + post(e, p)$.

□

Proposition 3.5 *$mc(\mathcal{M}, w, \Phi)$ is executed in polynomial time in the size of the input $(\mathcal{M}, w, \Phi)$.*

Proof The time is bounded the height of the computation tree rooted in $mc(\mathcal{M}, w, \Phi)$. Thanks to Lemma 3.4, the height of the computation treeis bounded by the quantity associated to $mc(\mathcal{M}\overrightarrow{\mathcal{E}}, w\overrightarrow{e}, \varphi)$, that is $|\mathcal{M}| + |\varphi|$. This quantity is *not* the size of the input $(\mathcal{M}, w, \Phi)$: for instance the weight of C_G-modalities is $\log_2 B_{\mathcal{M},\Phi}$. However this quantity is polynomial in the the size of the input $(\mathcal{M}, w, \Phi)$.

At each node of the computation tree, the computation performed in a single node is polynomial. For instance, the instruction '**(∀)** choose $u\overrightarrow{f} \in \mathcal{M}\overrightarrow{\mathcal{E}}$' consists in choosing each bit of $u\overrightarrow{f}$, thus is polynomial in the size of the input.

To conclude, the execution time on each branch in the computation tree is polynomial. $\square$

4 Upper bound of SAT

The satisfiability problem for **DELCK**, given a **DELCK**-formula Φ, asks to decide Φ is satisfiable. In this section, we prove the following upper bound result:

Theorem 4.1 *The satisfiability problem of* **DELCK** *is in* 2-EXPTIME.

In order to prove Theorem 4.1, we will proceed as for proving that Propositional Dynamic Logic is in EXPTIME and use the method of Pratt [22], but we will simulate tableau method rules of the same kind that in [1]. To ease the reading, we will w.l.o.g consider that formulas are in negative normal form, that is, negations are in front of atomic propositions, and we will use all connectives $\vee$, $\wedge$, K_a, $\hat{K}_a$, C_a, $\hat{C}_a$, $\langle e \rangle$, $[e]$. The *negation* of a formula φ is the formula in negative normal form obtained by negating all connectives, e.g. the negation of $C_G((\hat{K}_a \neg q) \wedge \langle e \rangle p)$ is formula $\hat{C}_G(K_a q \vee [e]\neg p)$. The dynamic modal depth of a formula Φ, noted $dmd(\Phi)$, is the modal depth by only counting the dynamic operators. E.g. the dynamic modal depth of $K_a[e][e']p \wedge [e']C_G q$ is 2.

Definition 4.2 The *closure* [2] of formula Φ is the set $Cl(\Phi)$ that contains elements $in_{\overrightarrow{e}}$ and $(\overrightarrow{e}, \psi)$ where $\overrightarrow{e}$ is a sequence of events in E of length at most $dmd(\Phi)$, and ψ is a subformula (or negation) of Φ or a subformula (or negation) of a precondition or postcondition in E, under the condition that $dmd(\varphi) + |\overrightarrow{e}| \leq dmd(\Phi)$.

The intended meaning of $in_{\overrightarrow{e}}$ is that the current world survives the sequence of events $\overrightarrow{e}$. The intended meaning of $(\overrightarrow{e}, \varphi)$ is that formula φ is true after having executed the sequence of events $\overrightarrow{e}$.

[2] The definition given here contains 'too many' formulas. We could have given a much more thorough definition, but the definition would have been more complicated to understand and the closure would have had the same asymptotic size.

Example 4.3 Let us take the event model $\mathcal{E}$ of Figure 2 and formula $\Phi := [e]K_a[f]q$. The closure $Cl(\Phi)$ is the set $\{in_\epsilon, in_e, in_{ef}, in_f, in_{fe}, (\epsilon, \Phi), (\epsilon, K_a[f]q), (e, K_a[f]q), \dots\}$.

Proposition 4.4 *The size of the closure of Φ is exponential in $|\Phi|$.*

Proof There is a direct correspondence between a subformula of Φ and a node in the syntactic tree of Φ. Therefore, the number of subformulas of Φ is in $O(|\Phi|)$. The number of possible ψ is then bounded by $O(|\Phi|)$ (the size of Φ is the number of memory cells needed to write down Φ, all the information of the event model $\mathcal{E}$ included). The number of possible sequences $\overrightarrow{e}$ is $|\mathsf{E}|^{dmd(\Phi)}$, thus exponential in $|\Phi|$. □

A Hintikka set (see Definition 4.5) is a maximal subset of $Cl(\Phi)$ that is consistent with respect to propositional logic (points 2-4), common knowledge reflexivity (point 5), dynamic operators (point 6-7), executability of events (point 8-9) and postconditions (point 10).

Definition 4.5 A *Hintikka set* h over $Cl(\Phi)$ is a subset of $Cl(\Phi)$ that satisfies:

(1)	If $(\overrightarrow{e}, \varphi) \in h$	then	$in_{\overrightarrow{e}} \in h$;
(2)	$(\overrightarrow{e}, \varphi \wedge \psi) \in h$	iff	$(\overrightarrow{e}, \varphi) \in h$ and $(\overrightarrow{e}, \psi) \in h$;
(3)	$(\overrightarrow{e}, \varphi \vee \psi) \in h$	iff	$(\overrightarrow{e}, \varphi) \in h$ or $(\overrightarrow{e}, \psi) \in h$;
(4)	If $in_{\overrightarrow{e}} \in h$	then	$(\overrightarrow{e}, \varphi) \in h$ xor $(\overrightarrow{e}, \neg\varphi) \in h$[3];
(5)	If $(\overrightarrow{e}, C_G\varphi) \in h$	then	$(\overrightarrow{e}, \varphi) \in h$;
(6)	$(\overrightarrow{e}, \langle \mathsf{E}_0 \rangle \varphi) \in h$	iff	there exists $e \in \mathsf{E}_0$ s.t. $in_{\overrightarrow{e}::e} \in h$ and $(\overrightarrow{e}::e, \varphi) \in h$;[4]
(7)	$(\overrightarrow{e}, [\mathsf{E}_0]\varphi) \in h$	iff	for all $e \in \mathsf{E}_0$, we have $in_{\overrightarrow{e}::e} \in h$ implies $(\overrightarrow{e}::e, \varphi) \in h$;
(8)	$in_\epsilon \in h$;		
(9)	$in_{\overrightarrow{e}::e} \in h$	iff	$in_{\overrightarrow{e}} \in h$ and $(\overrightarrow{e}, pre(e)) \in h$;
(10)	$(\overrightarrow{e}::e, p) \in h$	iff	$(\overrightarrow{e}, post(e)(p)) \in h$.

Point (1) means that if a Hintikka set contains $(\overrightarrow{e}, \varphi)$, then it means that $\overrightarrow{e}$ should be executable (in the intuitive world represented by the Hintikka set). Point (4) means that Hintikka sets are consistent. Point (5) says that if φ is common knowledge then φ is true. Points (6) and (7) mimics the truth condition given in Definition 2.6. Point (8) means the empty sequence of events ϵ is always executable. Point (9) means that $\overrightarrow{e}::e$ is executable iff $\overrightarrow{e}$ is executable and the precondition of e holds after having executed $\overrightarrow{e}$. Point (10) means that the truth of atomic proposition p after a non-empty sequence $\overrightarrow{e}::e$ of events is given by the truth of its postcondition before the last event e.

Now, we define the following structure that takes care of the consistency of the box modalities K_a, C_G.

[4] Formula $\neg\varphi$ is the negation of φ in the following sense: the negative normal obtained by negating all connectives in φ.

[4] We explicitly mentioned $in_{\overrightarrow{e}::e} \in h$ for uniformity with the semantics. However, note that it is implied by point (1).

```
function isDELCK-sat?($\Phi$)
    Compute the Hintikka structure $\mathcal{H} := (H, (R_a)_{a \in Ag})$ for $\Phi$
    repeat
        Remove any Hintikka set $h$ from $\mathcal{H}$ if
        ($\hat{K}_a$) either there is $(\overrightarrow{e}, \hat{K}_a\psi) \in h$ but no $h' \in R_a(h)$ with
              $(\overrightarrow{e}', \psi) \in h$ with $\overrightarrow{e} \rightarrow^a \overrightarrow{e}'$ and $in_{\overrightarrow{e}'} \in h'$;
        ($\hat{C}_G$) or there is $(\overrightarrow{e}, \hat{C}_G\psi) \in h$ but no path $h = h_0 \rightarrow^{a_1} h_1 \dots h_k$
              and no path $\overrightarrow{e} = \overrightarrow{e}^{(0)} \rightarrow^{a_1} \overrightarrow{e}^{(1)} \dots \rightarrow^{a_k} \overrightarrow{e}^{(k)}$ such that
              $(\overrightarrow{e}^{(k)}, \psi) \in h_k$ and $a_1, \dots, a_k \in G$ and $in_{\overrightarrow{e}^{(i)}} \in h_i$.
    until no more Hintikka sets are removed
    if there is still a Hintikka set in $\mathcal{H}$ containing $(\epsilon, \Phi)$ then accept
    else reject
endFunction
```

Figure 4. Algorithm for the satisfiability problem of a **DELCK**-formula Φ.

Definition 4.6 The *Hintikka structure* for φ is $\mathcal{H} := (H, (R_a)_{a \in Ag})$ where:

- H is the set of all possible Hintikka sets over $Cl(\Phi)$;
- hR_ah' if the two following conditions holds:

(K_a) for all $(\overrightarrow{e}, K_a\varphi) \in h$ we have $(\overrightarrow{e}', \varphi) \in h'$ for all $\overrightarrow{e}'$ such that $\overrightarrow{e} \rightarrow^a \overrightarrow{e}'$ and $in_{\overrightarrow{e}'} \in h'$,

(C_G) for all $(\overrightarrow{e}, C_G\varphi) \in h$ we have $(\overrightarrow{e}', C_G\varphi) \in h'$ for all $\overrightarrow{e}'$ such that $\overrightarrow{e} \rightarrow^a \overrightarrow{e}'$ with $a \in G$ and $in_{\overrightarrow{e}'} \in h'$.

The size of the Hintikka structure is double-exponential in $|\varphi|$, since there are a double-exponential number of different Hintikka sets. We finish by giving the algorithm isDELCK-sat? (see Figure 4) whose **repeat...until** loop takes care of the consistency of diamond modalities, $\hat{K}_a, \hat{C}_a$. The algorithm starts with the full Hintikka structure. Points ($\hat{K}_a$), ($\hat{C}_a$) remove worlds where $\hat{K}_a\psi$ and $\hat{C}_a\psi$ have no appropriate ψ-successor. We write $\overrightarrow{e} \rightarrow^a \overrightarrow{e}'$ if for all $(\overrightarrow{e}_i, \overrightarrow{e}'_i) \in R_a^{\mathcal{E}}$. Actually, the algorithm decides in double-exponential time whether a **DELCK**-formula is satisfiable (Propositions 4.7 and 4.8).

Proposition 4.7 *Algorithm isDELCK-sat? of Figure 4 runs in double-exponential time in* $|\Phi|$.

Proof The computation of $\mathcal{H}$ can be performed by brute-force: enumerate all subsets of $Cl(\Phi)$ and discard those which do not satisfy all conditions (1)-(10) of Definition 4.5. Compute R_a according to Definition 4.6. The loop is repeated at most the number of Hintikka sets in $\mathcal{H}$, that is $O(2^{2^{|\Phi|}})$ times, since at least one Hintikka set is removed or we exit the loop. Both tests ($\hat{K}_a$) and ($\hat{C}_G$) can be performed by depth-first search algorithm running in polynomial time in the size of the graph, that is of size double-exponential in $|\Phi|$. □

Proposition 4.8 Φ *is* **DELCK**-*satisfiable iff isDELCK-sat? accepts* Φ.

Proof ($\Rightarrow$) Let $\mathcal{M}, w$ such that $\mathcal{M}, w \models \Phi$. Given a world u, we note $h(u)$

the Hintikka set obtained by taking $in_{\overrightarrow{e}}$ if $\overrightarrow{e}$ is executable in u and $(\overrightarrow{e}, \psi)$ if ψ holds in $u, \overrightarrow{e}$. We show that no Hintikka set $h(u)$ is removed from $\mathcal{H}$. In particular, $h(w)$ is not removed, and contains (ϵ, Φ) so the algorithm isDELCK-sat? accepts Φ.

($\Leftarrow$) Suppose isDELCK-sat? accepts Φ. We construct a model $\mathcal{M} = (W, (R_a)_{a \in Ag}, V)$ as follows:

- W is the set of Hintikka sets that remain in the structure at the end of the algorithm;
- R_a is the relation for agent a at the end of the algorithm;
- $V(h) = \{p \in AP \mid (\epsilon, p) \in h\}$.

The proof finishes by proving the following lemma:

Lemma 4.9 *(truth lemma) The properties $\mathcal{P}(in_{\overrightarrow{e}})$ and $\mathcal{P}((\overrightarrow{e}, \varphi))$ defined below hold:*

- *$\mathcal{P}(in_{\overrightarrow{e}})$: for all $h \in W$, $in_{\overrightarrow{e}} \in h$ iff $\overrightarrow{e}$ is executable in $\mathcal{M}, h$;*
- *$\mathcal{P}((\overrightarrow{e}, \varphi))$: for all $h \in W$, $(\overrightarrow{e}, \varphi) \in h$ iff $\mathcal{M} \otimes \mathcal{E}^{|\overrightarrow{e}|}, (h, \overrightarrow{e}) \models \varphi$.*

Proof The proof is performed by induction by assigning the following quantities: the quantity for $in_{\overrightarrow{e}}$ is $n|\mathcal{E}|$; the quantity for $(\overrightarrow{e}, \varphi)$ is $n|\mathcal{E}| + |\varphi|$ where n is the length of $\overrightarrow{e}$, $|\mathcal{E}$ and $|\varphi|$ are defined as in Definition 3.3, except that now we use the traditional clause $|C_G\varphi| := |\varphi| + 1$.

□

We conclude by applying the truth lemma (Lemma 4.9 to the Hintikka set h that contains (ϵ, Φ) and we obtain that $\mathcal{M}, h \models \Phi$. □

Remark 4.10 The 2EXPTIME upper bound also holds for the satisfiability problem of **DELCK** in **S5** Kripke models. We proceed as in [15] (p. 358). We add the following clauses to definition 4.5:

$$\text{(5')} \quad \text{If } (\overrightarrow{e}, K_a\varphi) \in h \quad \text{then} \quad (\overrightarrow{e}, \varphi) \in h;$$

We add the following clause in the definition of R_a in Definition 4.6:

$$\text{for all } \overrightarrow{e} \to^a \overrightarrow{e}', \text{ if } in_{\overrightarrow{e}} \in h \text{ and } in_{\overrightarrow{e}'} \in h' \text{ then } (\overrightarrow{e}, K_a\varphi) \in h \text{ iff } (\overrightarrow{e}', K_a\varphi) \in h'.$$

5 Lower bound of SAT

5.1 Reduction

The aim of this section is to prove the following theorem.

Theorem 5.1 *The satisfiability problem of* **DELCK** *is* 2-EXPTIME-*hard.*

Let us consider any 2-EXPTIME decision problem L. As AEXPSPACE = 2-EXPTIME [8], it is decided by an alternating Turing machine M that runs in exponential space. W.l.o.g we suppose that all executions halt [5] and no state

[5] If not, we add a double exponential counter to the machine and we abort the execution after a double exponential number of steps.

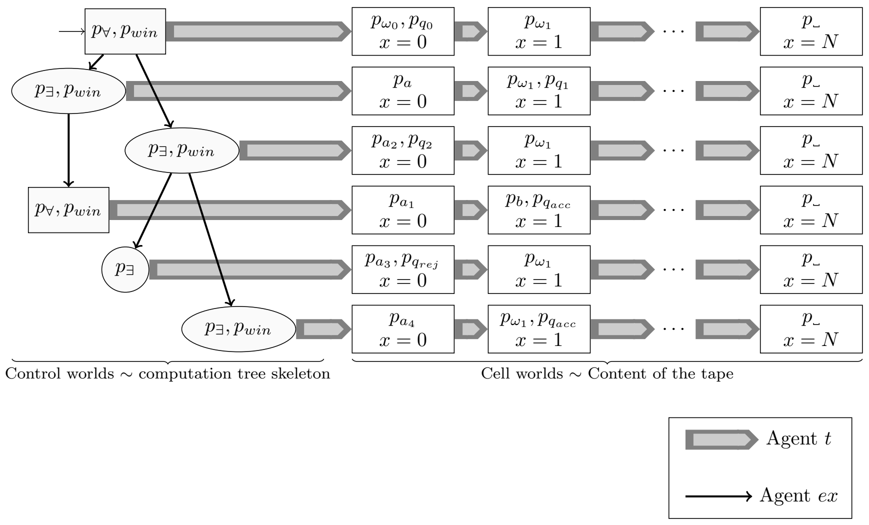

Figure 5. (Expected) Kripke model that represents the computation tree of M on the input instance ω.

is a negated state. We will define a polynomial reduction tr from L to the satisfiability problem of **DELCK**, that is tr will be computable in polynomial time, and ω is a positive instance of L if and only if $tr(\omega)$ is a satisfiable **DELCK**-formula.

The idea of $tr(\omega)$ is to enforce an expected form of a Kripke model as shown in Figure 5 that represents the computation tree of M starting with ω on the tape. The cursor of the machine remains in the N-first cell portion of the tape, where N is exponential in $|\omega|$. We define $N_0 = \log_2(N)$ for the rest of the section. N_0 is polynomial in $|\omega|$.

We introduce two agents: agent ex for the transitions in the computation tree and agent t for the linear structure of tapes. A configuration of the Turing machine is represented by a sequence of worlds linked by agent t: one so-called *control world* followed by *cell worlds*.

- The control world contains the type of the configuration: existential (resp. universal) if $p_\exists$ (resp. $p_\forall$) is true. A special atomic proposition p_{win} tags control worlds that correspond to winning configurations for player $\exists$.
- Cell worlds represent the cells of the tape and form a linear structure. They are indexed by x from $x = 0$ (left-most cell) to $x = N$ (right-most cell). In each cell world, p_a is true means that the corresponding cell contains letter $a \in \Gamma$. A proposition of the form p_q being true means that the cursor is at that cell and the current state is $q \in Q$.

Besides atomic propositions $p_\exists, p_\forall$, $p_a, a \in \Gamma$ and $p_q, q \in Q$, we also consider the list of atomic propositions for the bits of the cell index x: $x_1, \ldots, x_{N_0}$. We also consider another such list for another cell index v: $v_1, \ldots, v_{N_0}$. The index

v will be used to compare cell worlds of tapes of a configuration and a successor configuration different tapes during transitions.

The definition of $tr(\omega)$ needs multi-pointed event models $(\mathcal{E}^i, \mathsf{E}_0^i)$ given in Figure 6 non-deterministically and publicly choose the i^{th} bit of value v. We also consider Boolean formulas $x \leq v$, $x = v$, $x = v - 1$ and finally $K_t x = x + 1$ (the value of $K_t x_0 \dots K_t x_{N_0}$ is equal to $x + 1$). We define the abbreviation $[\mathtt{choosev}] = [\mathcal{E}^0, \mathsf{E}_0^0] \dots [\mathcal{E}^{N_0}, \mathsf{E}_0^{N_0}]$. Technically, it corresponds to non-deterministically choosing and publicly announcing a value for v.

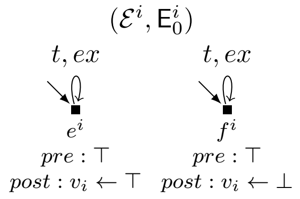
$(\mathcal{E}^i, \mathsf{E}_0^i)$
t, ex
t, ex
e^i
f^i
$pre : \top$
$pre : \top$
$post : v_i \leftarrow \top$
$post : v_i \leftarrow \bot$

Figure 6: Multi-pointed event models $(\mathcal{E}^i, \mathsf{E}_0^i)$.

Definition 5.2 Formula $tr(\omega)$ is the conjunction of the formulas shown in Table 1.

In formulas of Table 1, common knowledge operators C_{ex} and $\hat{C}_{ex}$ are used to talk about any control world, while C_t and $\hat{C}_t$ talk about any cell world.

Importantly, notice that with **DELCK**, it is impossible to force the each world to have exactly one successor. Thus in general, the expected Kripke model is not as depicted in Figure 5. Instead, we ensure that cell worlds of depth k have the value $x = k$. We use formula (ix) that imposes the value of x to be the same in all successor cell worlds, and formula (x) saying that everywhere, $K_t x = x + 1$. Formula (xi) states that only one p_a is true in each cell world, formula (xii) states that only one p_q is true in some cell world, and formula (xiii) states that if p_q is true in a cell world, then no p'_q is true in all the t-successors. Formulas (xvii), (xviii) and (xix) define the initial tape. Transitions are ensured by formulas (xiv) to (xvi). These formulas automatically ensure that several cell worlds with the same index x have the same valuation over $x, p_a, a \in \Gamma, p_q, q \in Q$.

Formulas that handle transitions use integer v to pinpoint a cell index in the tape. It is used in formula (xiv) to tell that when the cursor is not in a cell world, then the letter remains the same during any transition. It is also used in formula (xv) to check the existence of all compatible transitions and in formula (xvi) to check that all successor control worlds and their tapes correspond to a transition.

Proposition 5.3 *$tr(\omega)$ is satisfiable if and only if ω is a positive instance of L.*

The lower bound given in Theorem 5.1 still holds for the variant of the satisfiability problem where we require the model to be **S5**, that is, epistemic relations, to be equivalence relations.

	Valuations for control worlds	
(i)	$C_{ex}(x = 0)$	$x = 0$ holds in all control worlds.
(ii)	$C_{ex}\ (p_\exists \leftrightarrow \bigvee_{q \mid g(q)=\exists} \hat{C}_t p_q)$ $\wedge\ \ (p_\forall \leftrightarrow \bigvee_{q \mid g(q)=\forall} \hat{C}_t p_q)$	$p_\forall$ and $p_\exists$ match the type of the state on the tape.
(iii)	$C_{ex}\left(\bigwedge_{a\in\Gamma} \neg p_a \wedge \bigwedge_{q\in Q} \neg p_q\right)$	Every p_a or p_q is false.
	Winning condition	
(iv)	$C_{ex}((\hat{C}_t p_{q_{acc}}) \rightarrow p_{win})$	If the current state is q_{acc} the world is marked as winning.
(v)	$C_{ex}((\hat{C}_t p_{q_{rej}}) \rightarrow \neg p_{win})$	If the current state is q_{rej} the world is marked as losing.
(vi)	$C_{ex}((p_\forall \wedge (C_t \neg p_{q_{acc}}))$ $\rightarrow (p_{win} \leftrightarrow K_{ex} p_{win}))$	If the current state is not q_{acc} and is universal, the world is marked as winning if all successor worlds are marked as winning.
(vii)	$C_{ex}((p_\exists \wedge (C_t \neg p_{q_{acc}}))$ $\rightarrow (p_{win} \leftrightarrow \hat{K}_{ex} p_{win}))$	If the current state is not q_{acc} and is existential, the world is marked as winning if one successor world is marked as winning.
	Tape	
(viii)	$C_{ex} K_t (x = 0)$	The cell index of the left-most cell is 0.
(ix)	$C_{ex} K_t C_t \left(\bigwedge_{i=0}^{N} (K_t x_i \vee K_t \neg x_i)\right)$	On any tape world, the value of x is the same in all successors
(x)	$C_{ex} K_t C_t (K_t x = x + 1)$	On any tape world, the value of x is incremented by 1 on all successors.
(xi)	$C_{ex} K_t C_t \left(\oplus_{a\in\Gamma} p_a\right)$	On any tape world, only one p_a is true and represent the current letter on the cell.
(xii)	$C_{ex} \hat{C}_t \left(\oplus_{q\in Q} p_q\right)$	On any tape, somewhere only one p_q is true
(xiii)	$C_{ex} C_t \bigwedge_{q\in Q} \left(p_q \rightarrow C_t \bigwedge_{q'\in Q} \neg p_{q'}\right)$	Anywhere, if p_q is true then no $p_{q'}$ is true anywhere on the rest of the tape.
	Transitions	
	We define here $\varphi_{(q,a,q',b,d)} = C_t((x = v \rightarrow p_b \wedge \neg p_q) \wedge (x = v + d \rightarrow p_{q'}))$	
(xiv)	$[\texttt{choosev}] C_{ex} \bigwedge_{a\in\Gamma} \hat{C}_t$ $\left(p_a \wedge \bigwedge_{q\in Q} \neg p_q \wedge x = v\right)$ $\rightarrow K_{ex} C_t (x = v \rightarrow p_a)$	On the tape, if no p_q is true and p_a is true, then at the same position on the successors' tapes, p_a is true.
(xv)	$\bigwedge_{(q,a,q',b,d)\in\delta} [\texttt{choosev}]$ $\Big(C_{ex}(\hat{C}_t(p_q \wedge p_a \wedge x = v)$ $\rightarrow \hat{K}_{ex} \varphi_{(q,a,q',b,d)})\Big)$	If there is a transition it must be present on the model.
(xvi)	$[\texttt{choosev}] \bigwedge_{a\in\Gamma} \bigwedge_{q\in Q} C_{ex}$ $\Big(\hat{C}_t(p_q \wedge p_a \wedge x = v)$ $\rightarrow K_{ex} \bigvee_{(q,a,q',b,d)\in\delta} \varphi_{(q,a,q',b,d)}\Big)$	In every world, any ex-successor must correspond to a transition.
	Initial configuration	
(xvii)	$\bigwedge_{i=0}^{\lvert\omega\rvert-1} C_t((x = i) \rightarrow p_{\omega(i)})$	The letters of the initial word are on the initial tape.
(xviii)	$C_t((x \geq \lvert\omega\rvert) \rightarrow p_{\_})$	Cells of index $\lvert\omega\rvert$ are blank.
(xix)	$K_t p_{q_0}$	Head in the left-most cell. Initially in the initial state.
(xx)	p_{win}	The initial control world is winning.

Table 1
Clauses of **DELCK**-formula $tr(\omega)$.

6 Succinct models for Dynamic Epistemic Logic

We define the succinct models for DEL, i.e. succinct Kripke models and event models. Such models were originally introduced in [9] and [10], but we propose a simplification of the definitions compared to their original definitions. In particular, the presentation of the new definitions of succinct Kripke models and succinct event models are neater: their sets of atomic propositions and not mixed as in [10].

6.1 Accessibility programs

Instead of describing the epistemic relations $R^a_{\mathcal{M}}$ and $R^{\mathcal{E}}_a$ respectively in Kripke models and event models in extension, we describe them in intention by using *accessibility programs.* Technically, we use Dynamic Logic with Propositional Assignments proposed by Herzig et al. ([3], [4],).

Definition 6.1 The syntax for accessibility programs is defined by the BNF $\pi ::= p{\leftarrow}\beta \mid \beta? \mid (\pi;\pi) \mid (\pi \cup \pi)$ where $p \in AP$, β is a Boolean formula.

Program $p{\leftarrow}\beta$ reads as "assign atomic proposition p to the truth value of β". Program $\beta?$ reads as "test β". Program $\pi_1;\pi_2$ reads as "execute π_1 then π_2". Program $(\pi_1 \cup \pi_2)$ reads as "either execute π_1 or π_2". We write $\texttt{assign}(p_1, \ldots, p_n) = (p_1{\leftarrow}\bot \cup p_1{\leftarrow}\top); \ldots; (p_n{\leftarrow}\bot \cup p_n{\leftarrow}\top)$ for the program setting arbitrary values to $p_1, \ldots, p_n$.

Definition 6.2 The semantics of π is the binary relation over valuations defined by induction on π as follows, where $\texttt{w}$ and $\texttt{u}$ are valuations:

- $\texttt{w} \xrightarrow{p\leftarrow\beta} \texttt{u}$ if ($\texttt{u} = \texttt{w}\backslash\{p\}$ and $\texttt{w} \not\models \beta$) or ($\texttt{u} = \texttt{w} \cup \{p\}$ and $\texttt{w} \models \beta$);
- $\texttt{w} \xrightarrow{\beta?} \texttt{u}$ if $\texttt{w} = \texttt{u}$ and $\texttt{w} \models \beta?$;
- $\texttt{w} \xrightarrow{\pi_1;\pi_2} \texttt{u}$ if there exists a valuation $\texttt{v}$ such that $\texttt{w} \xrightarrow{\pi_1} \texttt{v}$ and $\texttt{v} \xrightarrow{\pi_2} \texttt{u}$;
- $\texttt{w} \xrightarrow{\pi_1\cup\pi_2} \texttt{u}$ if $\texttt{w} \xrightarrow{\pi_1} \texttt{u}$ or $\texttt{w} \xrightarrow{\pi_2} \texttt{u}$;

6.2 Succinct Kripke models

From now on, we suppose that we have a set AP to define the formulas.

Definition 6.3 A *succinct Kripke model* is a tuple $\mathfrak{M} = \langle AP_{\mathcal{M}}, \beta_{\mathcal{M}}, (\pi_a)_{a\in Ag}\rangle$ where $AP_{\mathcal{M}} \supseteq AP$ is a finite set of atomic propositions, $\beta_{\mathcal{M}}$ is a Boolean formula over $AP_{\mathcal{M}}$, and π_a is a program over $AP_{\mathcal{M}}$ for each agent a.

The Boolean formula β_M succinctly describes the set of epistemic states. Intuitively, each π_a succinctly describes the accessibility relation $\rightarrow_a$ for an agent a. A pointed succinct Kripke model is a pair $\mathfrak{M}, \texttt{w}$ where $\mathfrak{M} = \langle AP_{\mathcal{M}}, \beta_{\mathcal{M}}, (\pi_a)_{a\in Ag}\rangle$ is a succinct Kripke model and $\texttt{w}$ is a valuation satisfying $\beta_{\mathcal{M}}$.

Definition 6.4 Given a succinct Kripke model $\mathfrak{M} = \langle AP_{\mathcal{M}}, \beta_{\mathcal{M}}, (\pi_a)_{a\in Ag}\rangle$, the Kripke model represented by $\mathfrak{M}$, noted $\hat{\mathcal{M}}(\mathfrak{M})$ is the model $\mathcal{M} = (W, (R_a)_{a\in Ag}, V)$ where $W = \{\texttt{w} \in \mathcal{V}(AP_{\mathcal{M}}) \mid \texttt{w} \models \beta_{\mathcal{M}}\}$; $R_a = \left\{(\texttt{w}, \texttt{u}) \in W^2 \mid \texttt{w} \xrightarrow{\pi_a} \texttt{u}\right\}$; $V(\texttt{w}) = \texttt{w}$.

Example 6.5 In the muddy children example [19], each child does not know whether she is muddy or not, but knows the muddiness of the other children. If m_a is a proposition for "child a is muddy", a succinct Kripke model is $\mathfrak{M} = \langle AP_{\mathcal{M}}, \beta_{\mathcal{M}}, (\pi_a)_{a \in Ag} \rangle$ with $AP_{\mathcal{M}} = \{m_a, a \in Ag\}$, $\beta_{\mathcal{M}} = \top$ and $\pi_a = \mathtt{assign}(m_a)$. This representation is polynomial in the number of agents $|Ag|$ whereas the non-succinct Kripke model has exponential size in $|Ag|$.

Any Kripke model can be represented as a succinct Kripke model of polynomial size in the worst case. To do so, we define a succinct Kripke model $\mathfrak{M}_{\mathcal{M}}$ representing the Kripke model $\mathcal{M}$ with respect to a set of propositions AP.

Definition 6.6 Let $\mathcal{M} = (W, (R_a)_{a \in Ag}, V)$ be a Kripke model. We define the *succinct Kripke model* $\mathfrak{M}_{\mathcal{M}} = \langle AP_{\mathcal{M}}, \beta_{\mathcal{M}}, (\pi_a)_{a \in Ag} \rangle$ where: $AP_{\mathcal{M}} = AP \cup \{p_w \mid w \in W\}$; $\beta_{\mathcal{M}} = \exists!(\{p_w \mid w \in W\}) \wedge \bigwedge_{w \in W} p_w \rightarrow desc(V(w))$; $\pi_a = \bigcup_{w R_a u} p_w?; \mathtt{assign}(AP_{\mathcal{M}}); p_u?$.

Example 6.7 The Kripke model $\mathcal{M}$ from Figure 2 is modeled by the succinct Kripke model $\mathfrak{M}_{\mathcal{M}} = \langle AP_{\mathcal{M}}, \beta_{\mathcal{M}}, (\pi_a)_{a \in Ag} \rangle$ with $AP_{\mathcal{M}} = \{p, p_w, p_u\}$, $\beta_{\mathcal{M}} = \exists!(\{p_u, p_w\}) \wedge (p_w \rightarrow p) \wedge (p_u \rightarrow \neg p)$ and $\pi_a = \bigcup_{w_1, w_2 \in W} p_{w_1}?; \mathtt{assign}(AP_{\mathcal{M}}); p_{w_2}?$.

6.3 Succinct event models

We define succinct event models in the same spirit than succinct Kripke models.

Definition 6.8 A *succinct event model* is a tuple $\mathfrak{E} = \langle AP_{\mathcal{E}}, \chi_{\mathcal{E}}, (\pi_{a,\mathcal{E}})_{a \in Ag}, \mathsf{pre}, \mathsf{post} \rangle$ where $AP_{\mathcal{E}}$ is a set of atomic propositions disjoint from AP; $\chi_{\mathcal{E}}$ is a propositional formula over $AP_{\mathcal{E}}$ characterizing the set of events; $\pi_{a,\mathcal{E}}$ is a program over $AP_{\mathcal{E}}$ for all $a \in Ag$; pre is a propositional formula over $AP_{\mathcal{E}} \cup \mathcal{L}_{\mathbf{EL}}(AP)$ (meaning that any atom from $AP_{\mathcal{E}}$ cannot be under the scope of a K or a C operator); For all $p \in AP$, $\mathsf{post}(p)$ is a propositional formula over $AP_{\mathcal{E}} \cup \mathcal{L}_{\mathbf{EL}}(AP)$.

Definition 6.9 Given a succinct event model $\mathfrak{E} = \langle AP_{\mathcal{E}}, \chi_{\mathcal{E}}, (\pi_{a,\mathcal{E}})_{a \in Ag}, \mathsf{pre}, \mathsf{post} \rangle$, the event model represented by $\mathfrak{E}$, noted $\hat{\mathcal{E}}(\mathfrak{E})$ is the model $(\mathsf{E}, (R_a^{\mathcal{E}})_{a \in Ag}, pre, post)$ on AP where $\mathsf{E} = \{\mathrm{v_e} \in \mathcal{V}(AP_{\mathcal{E}}) \mid \mathrm{v_e} \models \chi\}$;$R_a^{\mathcal{E}} = \{(\mathrm{v_e}, \mathrm{v_e}') \mid \mathrm{v_e} \xrightarrow{\pi_{a,\mathcal{E}}} \mathrm{v_e}'\}$; $pre(\mathrm{v_e}) = \mathsf{pre} \wedge desc(\mathrm{v_e})$; $post(\mathrm{v_e}, p) = \mathsf{post}(p) \wedge desc(\mathrm{v_e})$.

Definition 6.10 Let $\mathcal{E} = (\mathsf{E}, (R_a^{\mathcal{E}})_{a \in Ag}, pre, post)$ be an event model on AP. We define the succinct event model $\mathfrak{E}_{\mathcal{E}} = \langle AP_{\mathcal{E}}, \chi_{\mathcal{E}}, (\pi_{a,\mathcal{E}})_{a \in Ag}, \mathsf{pre}, \mathsf{post} \rangle$ where $AP_{\mathcal{E}} = \{p_e \mid e \in \mathsf{E}\}$; $\chi_{\mathcal{E}} = \exists!(AP_{\mathcal{E}})$; $\pi_{a,\mathcal{E}} = \bigcup_{e R_a^{\mathcal{E}} f} p_e?; p_e \leftarrow \bot; p_f \leftarrow \top$; $\mathsf{pre} = \bigwedge_{e \in \mathsf{E}} (p_e \rightarrow pre(e))$; $\mathsf{post}(p) = \bigwedge_{e \in \mathsf{E}} (p_e \rightarrow post(e, p))$.

6.4 Complexity of decision problems

Naturally, the pointed Kripke model is replaced by a pointed succinct Kripke model in the model checking. Formulas contain dynamic modalities $\langle \mathfrak{E}, \beta_0 \rangle$ where $\mathfrak{E}, \beta_0$ is a pointed succinct event model, instead of $\langle \mathcal{E}, \mathsf{E}_0 \rangle$. This new language is called $\mathcal{L}^{suc}_{\mathbf{DELCK}}$. We translate a succinct formula $\varphi \in \mathcal{L}^{suc}_{\mathbf{DELCK}}$ into a formula $\tau(\varphi)$ as follows:

- $\tau(\langle \mathfrak{E}, \beta_0 \rangle \varphi) := \langle \hat{\mathcal{E}}(\mathfrak{E}), \{\mathtt{v}_\mathrm{e} \in \mathcal{V}(AP_\mathcal{E}) \mid \mathtt{v}_\mathrm{e} \models \beta_0\} \rangle \tau(\varphi)$.

Interestingly, the upper complexities of both the model checking problem and the satisfiability problem remain the same in the succinct case: the core reason is that $\underbrace{2^{poly(n)} \times \dots 2^{poly(n)}}_{n \text{ times}} = 2^{poly(n)}$. Technical details are omitted due to space restriction.

7 Conclusion

Complexity results for dynamic epistemic logic with common knowledge was left open since the first complexity results in 2013 [1]. In this paper, we proved that the model checking of **DELCK** remains in PSPACE, even for succinct models. This result somehow justifies that BDD techniques are applicable for solving the model checking problem of **DELCK** in practice, as done in the tool DEMO [25].

We proved that the satisfiability problem of **DELCK** is 2-EXPTIME-hard. Actually, we only need trivial preconditions, Boolean postconditions and multi-pointed event models to obtain this lower bound. As a direct corollary, it implies that the satisfiability problem of the logic defined in [30] that contains knowledge, common knowledge operators, public assignments and non-deterministic $\cup$ over programs is also 2-EXPTIME-hard. The fall in PSPACE of the logic of public announcement and public assignment given in [28] is to due the absence of the common knowledge operator in their specification language. There is a long avenue of research to classify fragments of **DELCK** and evaluate the exact complexity of them. For instance, the exact of complexity of **DELCK** where event models are non-ontic (no postconditions) is an open question.

We also proved that the satisfiability problem of **DELCK** is in 2-EXPTIME. The proof technique is an adaptation of Pratt's technique for proving that **PDL** is in EXPTIME. Actually **DELCK** required such a deep machinery and more direct proofs (for instance via reduction axioms [26]) were not successful. We could infer a tableau method from our adaptation of Pratt's technique. We claim that one of the tableau rule requires an unbounded number of non-deterministic choices (actually exponential in the size of the input formula). Unfortunately, as far as we know, generic available tableau method provers, as Mettel2 [24] or Lotrec [11], only allows for a fixed number of non-deterministic choice in a given rule and do not provide any mechanism for allowing tableau rules with an unbounded number of non-deterministic choices.

Acknowledgments. We would like to thank Andreas Herzig, Renate A. Schmidt and Dmitry Tishkovsky for email discussions about tableau provers, and the anonymous reviewers for their useful comments.

References

[1] Aucher, G. and F. Schwarzentruber, *On the complexity of dynamic epistemic logic*, in: *Proceedings of the 14th Conference on Theoretical Aspects of Rationality and Knowledge (TARK 2013), Chennai, India*, 2013.
URL `http://www.tark.org/proceedings/tark_jan7_13/p19-aucher.pdf`

[2] Aumann, R. J., *Interactive epistemology I: Knowledge*, Int. J. Game Theory **28** (1999), pp. 263–300.
URL `http://dx.doi.org/10.1007/s001820050111`

[3] Balbiani, P., A. Herzig, F. Schwarzentruber and N. Troquard, *DL-PA and DCL-PC: Model checking and satisfiability problem are indeed in PSPACE*, CoRR **abs/1411.7825** (2014).
URL `http://arxiv.org/abs/1411.7825`

[4] Balbiani, P., A. Herzig and N. Troquard, *Dynamic logic of propositional assignments: A well-behaved variant of PDL*, in: *LICS*, 2013, pp. 143–152.

[5] Bolander, T. and M. B. Andersen, *Epistemic planning for single and multi-agent systems*, Journal of Applied Non-Classical Logics **21** (2011), pp. 9–34.
URL `http://dx.doi.org/10.3166/jancl.21.9-34`

[6] Bolander, T., M. H. Jensen and F. Schwarzentruber, *Complexity results in epistemic planning*, in: *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina*, 2015, pp. 2791–2797.
URL `http://ijcai.org/Abstract/15/395`

[7] Chandra, A. K., D. Kozen and L. J. Stockmeyer, *Alternation*, J. ACM **28** (1981), pp. 114–133.
URL `http://doi.acm.org/10.1145/322234.322243`

[8] Chandra, A. K. and L. J. Stockmeyer, *Alternation*, in: *Foundations of Computer Science, 1976., 17th Annual Symposium on*, IEEE, 1976, pp. 98–108.

[9] Charrier, T. and F. Schwarzentruber, *Arbitrary public announcement logic with mental programs*, in: *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2015, Istanbul, Turkey*, 2015, pp. 1471–1479.
URL `http://dl.acm.org/citation.cfm?id=2773340`

[10] Charrier, T. and F. Schwarzentruber, *A succinct language for dynamic epistemic logic*, in: *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems, AAMAS 2017, São Paulo, Brazil*, 2017, pp. 123–131.
URL `http://dl.acm.org/citation.cfm?id=3091148`

[11] del Cerro, L. F., D. Fauthoux, O. Gasquet, A. Herzig, D. Longin and F. Massacci, *Lotrec: The generic tableau prover for modal and description logics*, in: *Automated Reasoning, First International Joint Conference, IJCAR 2001, Siena, Italy, Proceedings*, 2001, pp. 453–458.
URL `https://doi.org/10.1007/3-540-45744-5_38`

[12] Engesser, T., T. Bolander, R. Mattmüller and B. Nebel, *Cooperative epistemic multi-agent planning with implicit coordination*, in: *Proceedings of the ICAPS-2015 Workshop on Distributed and Multi-Agent Planning (DMAP 2015)*, 2015.

[13] Gasquet, O., V. Goranko and F. Schwarzentruber, *Big brother logic: Visual-epistemic reasoning in stationary multi-agent systems*, Autonomous Agents and Multi-Agent Systems **30** (2016), pp. 793–825.
URL `https://doi.org/10.1007/s10458-015-9306-4`

[14] Halpern, J. Y. and Y. Moses, *Knowledge and common knowledge in a distributed environment*, J. ACM **37** (1990), pp. 549–587.
URL `http://doi.acm.org/10.1145/79147.79161`

[15] Halpern, J. Y. and Y. Moses, *A guide to completeness and complexity for modal logics of knowledge and belief*, Artif. Intell. **54** (1992), pp. 319–379.
URL `https://doi.org/10.1016/0004-3702(92)90049-4`

[16] Halpern, J. Y. and M. Y. Vardi, *Model checking vs. theorem proving: A manifesto*, in: *Proceedings of the 2nd International Conference on Principles of Knowledge Representation and Reasoning (KR'91). Cambridge, MA, USA*, 1991, pp. 325–334.

[17] Kozen, D., “Theory of Computation,” Texts in Computer Science, Springer, 2006.
URL `https://doi.org/10.1007/1-84628-477-5`

[18] Ladner, R. E., *The computational complexity of provability in systems of modal propositional logic*, SIAM J. Comput. **6** (1977), pp. 467–480.
URL `https://doi.org/10.1137/0206033`

[19] McCarthy, J., *Formalization of two puzzles involving knowledge*, Unpublished note, Stanford University (1978).

[20] Meyer, J.-J. C. and W. Van Der Hoek, “Epistemic Logic for AI and Computer Science” **41**, Cambridge University Press, 1995.

[21] Papadimitriou, C. H., “Computational Complexity,” Academic Internet Publ., 2007.

[22] Pratt, V. R., *A near-optimal method for reasoning about action*, J. Comput. Syst. Sci. **20** (1980), pp. 231–254.
URL `http://dx.doi.org/10.1016/0022-0000(80)90061-6`

[23] Schnoebelen, P., *The complexity of temporal logic model checking*, in: *Advances in Modal Logic 4, papers from the fourth conference on "Advances in Modal logic," held in Toulouse (France)*, 2002, pp. 393–436.

[24] Tishkovsky, D., R. A. Schmidt and M. Khodadadi, *The tableau prover generator mettel2*, in: *Logics in Artificial Intelligence - 13th European Conference, JELIA 2012, Toulouse, France*, 2012, pp. 492–495.
URL `https://doi.org/10.1007/978-3-642-33353-8_41`

[25] van Benthem, J., J. van Eijck, M. Gattinger and K. Su, *Symbolic model checking for dynamic epistemic logic*, in: *Logic, Rationality, and Interaction - 5th International Workshop, LORI 2015 Taipei, Taiwan*, 2015, pp. 366–378.
URL `https://doi.org/10.1007/978-3-662-48561-3_30`

[26] van Benthem, J., J. van Eijck and B. P. Kooi, *Logics of communication and change*, Inf. Comput. **204** (2006), pp. 1620–1662.
URL `http://dx.doi.org/10.1016/j.ic.2006.04.006`

[27] van de Pol, I., I. van Rooij and J. Szymanik, *Parameterized complexity results for a model of theory of mind based on dynamic epistemic logic*, in: *Proceedings Fifteenth Conference on Theoretical Aspects of Rationality and Knowledge, TARK 2015, Carnegie Mellon University, Pittsburgh, USA*, 2015, pp. 246–263.
URL `https://doi.org/10.4204/EPTCS.215.18`

[28] van Ditmarsch, H., A. Herzig and T. D. Lima, *Public announcements, public assignments and the complexity of their logic*, Journal of Applied Non-Classical Logics **22** (2012), pp. 249–273.
URL `https://doi.org/10.1080/11663081.2012.705964`

[29] Van Ditmarsch, H., W. van Der Hoek and B. Kooi, “Dynamic Epistemic Logic” **337**, Springer Science & Business Media, 2007.

[30] van Ditmarsch, H. P., W. van der Hoek and B. P. Kooi, *Dynamic epistemic logic with assignment*, in: *4th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2005), Utrecht, The Netherlands*, 2005, pp. 141–148.
URL `http://doi.acm.org/10.1145/1082473.1082495`