A Succinct Language for Dynamic Epistemic Logic

Tristan Charrier
IRISA
263 Avenue du General Leclerc - CS 74205
35042 Rennes Cedex, France
tristan.charrier@irisa.fr

Francois Schwarzentruber
ENS Rennes
Avenue Robert Schuman
35170 Bruz, France
francois.schwarzentruber@ens-rennes.fr

ABSTRACT

Dynamic epistemic logic (DEL) is an extension of modal multi-agent epistemic logic with dynamic operators. We propose a succinct version of DEL where Kripke models and event models are described succinctly. Our proposal relies on Dynamic logic of propositional assignments (DLPA): epistemic relations are described with so-called accessibility programs written in DLPA. We give examples of models that are exponentially more succinct in our framework. Interestingly, the model checking of DEL is PSPACE-complete and we show that it remains in PSPACE for the succinct version.

Keywords

Dynamic epistemic logic, propositional assignment, model checking, succinctness, complexity.

1. INTRODUCTION

Agents take decisions according to their knowledge about the world and their knowledge about other agents' knowledge (higher-order knowledge). Dynamic epistemic logic ([4], [16]) is a framework designed for expressing higher-order knowledge properties and dynamics.

Actions in dynamic epistemic logic are described by means of graphs called *event models*. Specific kinds of actions have been considered in the literature For instance, public announcements are represented by single-node event models.

Attention-based announcements [7] where some agents may listen to the source or not can be represented too by event models, but their sizes are exponential in the number of agents in the system.

However, we claim that this exponential blow-up in the representation is artificial and that is why we address succinctness in dynamic epistemic logic.

Usually, if the description language is (exponentially) more succinct, algorithmic problems become (exponentially) harder. For instance, deciding the existence of an Hamiltonian cycle is NP-complete but it becomes NEXPTIME-complete [13] when the input graph is described succinctly. A succinct representation of a graph with 2^b nodes is a Boolean circuit C such that there is an edge $(i,j) \in \{0,\dots,2^b-1\}^2$ iff C accepts the binary representations of the b-bit integers i,j as inputs.

In dynamic epistemic logic, the results are surprising. Whereas the model checking of DEL is PSPACE-complete [1], we would expect that the succinct version of the model checking is EXPSPACE-complete. Actually, we provide a framework where the representation of actions such as attention-based announcements is exponentially more succinct whereas the model checking remains in PSPACE.

In our framework, we do not use Boolean circuits traditionally used for representing instances for succinct decision problems (see [13], Chapter 20.1) but *accessibility programs* based on Dynamic Logic of Propositional Assignments (DLPA) ([3], [2]) as developed in [11], where they were called mental programs. The reason of that choice is that our model checking algorithm directly relies on DLPA. It extends the language of Van Benthem *et al.* [14] by including DLPA programs and postconditions for event models.

After having recalled some background on Dynamic epistemic logic in Section 2, the main contribution is to provide a succinct version of DEL in Section 3: we provide succinct models for DEL, prove that DEL can be embedded in succinct DEL (and *vice versa*), and prove that succinct DEL is exponentially more succinct than DEL. The succinctness of succinct Kripke models is rather easy to prove but the proof of the succinctness of succinct event models relies on the definition of *action emulations* [18]. In Section 4, we prove that the model checking problem in succinct DEL is (still) in PSPACE. More details may be found in [12].

2. BACKGROUND ON DEL

We first start by some notations about valuations, then present Dynamic epistemic logic (DEL). Let AP be a finite set of atomic propositions and Ag be a finite set of agents.

2.1 Valuations

The set of valuations over AP is denoted by $\mathcal{V}(AP)$. Valuations are denoted $\mathtt{w}, \mathtt{u}, \dots$ and they are represented by the set of true atomic propositions. The restriction of $\mathtt{w}$ to a subset of propositions AP' is $\mathtt{w} \cap AP'$ and is noted $\mathtt{w}_{|AP'}$. For any valuation $\mathtt{w}$ over AP, we note $desc(\mathtt{w})$ the formula $\bigwedge_{p\in\mathtt{w}} p \wedge \bigwedge_{p\in AP\setminus\mathtt{w}} \neg p$ that describes the valuation $\mathtt{w}$ (AP is made implicit for simplifying the notation). For a set AP, we note $\exists!(AP)$ the formula $\bigvee_{p\in AP}\left(p \wedge \bigwedge_{q\in AP|q\neq p} \neg q\right)$. The size of a valuation defined over AP is the cardinal of AP, noted $|AP|$ (we implement such a valuation by a bit array of size AP).

Appears in: *Proc. of the 16th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2017), S. Das, E. Durfee, K. Larson, M. Winikoff (eds.), May 8–12, 2017, São Paulo, Brazil.*
Copyright © 2017, International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). All rights reserved.

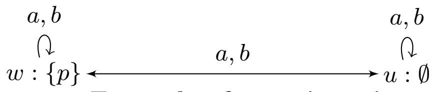


Figure 1: Example of an epistemic model

2.2 Epistemic models

Kripke models represent static epistemic situations and are defined as follows.

Definition 1. *A Kripke model $M = (W, (\rightarrow_a)_{a \in Ag}, V)$ is defined by a non-empty set W of epistemic states/worlds, epistemic relations $(\rightarrow_a)_{a \in Ag} \subseteq W \times W$ and a valuation function $V : W \rightarrow 2^{AP}$.*

Typically, W represent all possible configurations. V is a labeling for the states for describing the configuration. The intuitive meaning of $w \rightarrow_a u$ is that agent a considers state u as possible when the actual state is w. We do not require $\rightarrow_a$ to be an equivalence relation. The size of a Kripke model M is implemented by a labeled graph and each relation $\rightarrow_a$ is represented by an adjacency list. The size of M is thus $O(|Ag| \times |W|^2 + |W| \times |AP|)$.

Example 1. *Figure 1 depicts the model $\mathcal{M} = (W, (\rightarrow_a)_{a \in Ag}, V)$ given by $W = \{w, u\}$, $\rightarrow_a = \rightarrow_b = W \times W$, $V(w) = \{p\}$ and $V(u) = \emptyset$. Agents a and b do not distinguish w from u, therefore they do not know the truth value of p (in both w and u).*

Example 2. *We consider the extension of the classical muddy children puzzle [15] where children may pay attention to the father or not (as in [7]). We introduce atomic proposition m_a meaning that a is muddy and atomic proposition h_a meaning that agent a hears (i.e. pays attention to) the announcements of the father. We consider the model $M = (W, (\rightarrow_a)_{a \in Ag}, V)$ where $W = \mathcal{V}(\{m_a, h_a \mid a \in Ag\})$, $\mathtt{w} \rightarrow_a \mathtt{u}$ if ($\mathtt{w} \models h_a$ iff $\mathtt{u} \models h_a$) and for all $b \neq a$ ($\mathtt{w} \models m_b$ iff $\mathtt{u} \models m_b$), and $V(\mathtt{w}) = \mathtt{w}$ for all $w \in W$.*

In M, an agent a will not distinguish two worlds $\mathtt{w}$ from $\mathtt{u}$ as long he sees the same forehead states for the other agents and his pay attention status is the same in both worlds.

A Kripke model represents the state of mind of agents. A pointed Kripke model is a pair (M, w) with $w \in W$, modeling the fact that the current epistemic state is w.

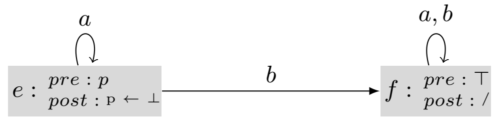
2.3 Syntax of epistemic language

The language $\mathscr{L}_{EL}$ extends the propositional language $\mathscr{L}_{Prop}$ with modal operators K_a and is defined by the following BNF: $\varphi ::= \top \mid p \mid \neg\varphi \mid (\varphi \vee \varphi) \mid K_a\varphi$, with $p \in AP$, $a \in Ag$. The formula $K_a\varphi$ is read "agent a knows that φ holds". We define the usual abbreviations $(\varphi_1 \wedge \varphi_2)$ for $\neg(\neg\varphi_1 \vee \neg\varphi_2)$ and $\hat{K}_a\varphi$ for $\neg K_a \neg\varphi$. The size of φ is the number of operators needed to write φ.

2.4 Event models

The dynamics of the system is modeled by event models.

Definition 2. *An* event model *$\mathcal{E} = (E, (\rightarrow_a^{\mathcal{E}})_{a \in Ag}, pre, post)$ is defined by a non-empty set of* events *E, epistemic relations $(\rightarrow_a^{\mathcal{E}})_{a \in Ag} \subseteq E \times E$, a precondition function $pre : E \rightarrow \mathscr{L}_{EL}$ and a postcondition function $post : E \times AP \rightarrow \mathscr{L}_{Prop}$.*



a
a, b
b
$e : \begin{matrix} pre : p \\ post : p \leftarrow \bot \end{matrix}$
$f : \begin{matrix} pre : \top \\ post : / \end{matrix}$

Figure 2: Example of an event model

The precondition function defines whether a given event is executable or not. The postcondition updates the values of atomic propositions after executing an event e: the truth of p is assigned to the value $post(e, p)$.

Remark 1. *Some authors ([17]) consider epistemic postcondition functions $post : E \times AP \rightarrow \mathscr{L}_{EL}$. Actually, it is always possible to compute an equivalent event model with a propositional postcondition function from an event model with an epistemic postcondition function. Such a transformation is given in [17] but is exponential. Fortunately, there exists a polynomial alternative transformation where an event model with epistemic postcondition functions is transformed into a sequence of event models with propositional postcondition functions*[1].

A event model is *without postconditions* (i.e. it does not change physically the current state) when $post(e, p) = p$ for all $e \in E$ and all atomic propositions p, that is when $post$ is *trivial*. In that case, we usually omit the $post$ function and we write $\mathcal{E} = (E, (\rightarrow_a^{\mathcal{E}})_{a \in Ag}, pre)$.

We introduce the notation $e \in \mathcal{E}$ for $e \in E$. A pointed event model is a pair $(\mathcal{E}, e)$ with $e \in E$. A multi-pointed event model is a pair $(\mathcal{E}, E_0)$ with $E_0 \subseteq E$. The size of $\mathcal{E}$ is similarly defined than the size of a Kripke model.

Example 3. *Figure 2 shows the event model $\mathcal{E} = (E, (\rightarrow_a^{\mathcal{E}})_{a \in Ag}, pre, post)$ where $E = \{e, f\}$, $\rightarrow_a^{\mathcal{E}} = \{(e, e), (f, f)\}$, $\rightarrow_b^{\mathcal{E}} = \{(f, f), (e, f)\}$, $pre(e) = p$, $pre(f) = \top$, $post(e, p) = \bot$ and $post(f, p) = p$.*

Example 4. *We focus on the notion of attention-based announcement of p as shown in [7]. In addition to classic atomic propositions, we add propositions h_a for "agent a is listening to the announcement". The attention-based announcement of p can be then represented by the event model $\mathcal{E} = (E, (\rightarrow_a^{\mathcal{E}})_{a \in Ag}, pre)$ where:*

- *$E = \mathcal{V}(\{p\} \cup \{h_a \mid a \in Ag\}) \cup \{idle\}$;*
- *for all a, $\mathtt{e} \rightarrow_a^{\mathcal{E}} \mathtt{f}$ if $\mathtt{e} \neq idle$, $\mathtt{f} \neq idle$, $\mathtt{e} \models h_a$, $\mathtt{f} \models p$; $\mathtt{e} \rightarrow_a^{\mathcal{E}} idle$ if $\mathtt{e} \neq idle$ and $\mathtt{e} \not\models h_a$; $idle \rightarrow_a^{\mathcal{E}} idle$;*
- *$pre(e) = desc(e)$ if $e \neq idle$, $\top$ otherwise.*

Figure 4 shows the event model of the attention-based announcement of p for two agents a_1 and a_2.

Event idle is the event where nothing happens. The relation $\rightarrow_a^{\mathcal{E}}$ is defined as follows: if a is listening ($\mathtt{e} \models h_a$) then a believes that p has been announced ($\mathtt{f} \models p$). If a is not listening ($\mathtt{e} \not\models h_a$) then a believes nothing happens. The precondition is defined to match the fact that attentive agents listen to the announcement of p (thus leading to events where p holds) and that other agents believe that nothing happened (thus the $\top$ precondition on idle). The announcement is purely epistemic so the postcondition function is trivial.

[1]The proof of this claim is in [12].

$$\begin{tikzcd}
(w,e) : \emptyset \arrow[loop above, "a"] \arrow[d, "b"'] \arrow[drr, "b"] & & \\
(w,f) : \{p\} \arrow[loop below, "{a,b}"] \arrow[rr, leftrightarrow, "{a,b}"'] & & (u,f) : \emptyset \arrow[loop below, "{a,b}"]
\end{tikzcd}$$

Figure 3: Example of a product

2.5 Product

The update of a Kripke model M with an event model $\mathcal{E}$ is defined by the synchronous product of both models, noted $M \otimes \mathcal{E}$, and defined as follows.

Definition 3. *Let $M = (W, (\rightarrow_a)_{a \in Ag}, V)$ be a Kripke model. Let $\mathcal{E} = (E, (\rightarrow_a^{\mathcal{E}})_{a \in Ag}, pre, post)$ be an event model. The* product *of M and $\mathcal{E}$ is $M \otimes \mathcal{E} = (W', (\rightarrow_a)', V')$ where:*

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

Example 5. *Figure 3 shows the product of the Kripke model of Figure 1 and the event model of Figure 2.*

2.6 Syntax

The language $\mathscr{L}_{DEL}$ extends $\mathscr{L}_{EL}$ and is defined by the following BNF.

$$\varphi \quad ::= \ \top \mid p \mid \neg\varphi \mid (\varphi \vee \varphi) \mid K_a\varphi \mid \langle \mathcal{E}, E_0 \rangle \varphi$$

with $p \in AP$, $a \in Ag$. Formula $\langle \mathcal{E}, E_0 \rangle \varphi$ is read "There exists an execution of $(\mathcal{E}, E_0)$ such that φ holds".

2.7 Semantics

We now define the semantics of a formula φ of $\mathscr{L}_{DEL}$.

Definition 4. *We define $M, w \models \varphi$ (φ is true in the pointed Kripke model M, w) by induction on φ:*

- *$M, w \models p$ iff $p \in V(w)$; $M, w \models \neg\varphi$ iff $\mathcal{M}, w \not\models \varphi$;*
- *$M, w \models (\varphi \vee \psi)$ iff $M, w \models \varphi$ or $M, w \models \psi$;*
- *$M, w \models K_a\varphi$ iff for all $u \in W$,*
  $w \rightarrow_a u$ implies $M, u \models \varphi$;
- *$M, w \models \langle \mathcal{E}, E_0 \rangle \varphi$ iff there exists $e \in E_0$ s.th.*
  $M, w \models pre(e)$ and $M \otimes \mathcal{E}, (w, e) \models \varphi$.

Example 6. *We consider the Kripke model M of Figure 1 and the event model of Figure 2. As $p \notin V(u)$ and $w \rightarrow_a u$, we have $M, w \models \neg K_a p$. As $\neg p$ holds in all $\rightarrow_a$-successors of (w, e) (Figure 3), $M \otimes \mathcal{E}, (w, e) \models K_a \neg p$, we have $M, w \models \langle \mathcal{E}, \{e\} \rangle K_a \neg p$.*

2.8 Bisimulation and equivalence

We define notions of equivalence for models.

2.8.1 Bisimulations for Kripke models

The usual notion of equivalence for Kripke models is bisimulation.

Definition 5. *Let AP be a finite set of atomic propositions, $M = (W, (\rightarrow_a)_{a \in Ag}, V)$ and $M' = (W', (\rightarrow'_a)_{a \in Ag}, V')$ be two Kripke models. A relation $B \subseteq W \times W'$ is a AP-bisimulation between M and M' iff for all $w \in W, w' \in W'$ such that wBw':*

- *Invariance: $V(w)_{|AP} = V'(w')_{|AP}$;*
- *Zig: for all $u \in W$ such that $w \rightarrow_a u$ there exists $u' \in W'$ such that $w' \rightarrow'_a u'$ and uBu';*
- *Zag: for all $u' \in W'$ such that $w' \rightarrow'_a u'$ there exists $u \in W$ such that $w \rightarrow_a u$ and uBu'.*

Two pointed Kripke models (M, w) and (M', w') are AP-bisimilar if there is a AP-bisimulation B between M and M' with wBw'. (M, w) and (M', w') are AP-bisimilar iff $((M, w) \models \varphi$ iff $(M', w') \models \varphi)$ for all formulas φ of $\mathscr{L}_{DEL}$, whose propositions are in AP. When AP is clear from the context, we say bisimilar instead of AP-bisimilar. Two Kripke models M and M' are bisimilar if there exists $w \in W$ and $w' \in W'$ such that (M, w) and (M', w') are bisimilar.

2.8.2 Equivalence of event models

For event models, the equivalence is defined as follows.

Definition 6. *Let $\mathcal{E}$ and $\mathcal{E}'$ be two pointed event models. They are equivalent if for all pointed Kripke models (M, w), for all $e \in \mathcal{E}$, there exists $e' \in \mathcal{E}'$ such that $(M \otimes \mathcal{E}, (w, e))$ and $(M \otimes \mathcal{E}', (w, e'))$ are bisimilar (and* vice versa*).*

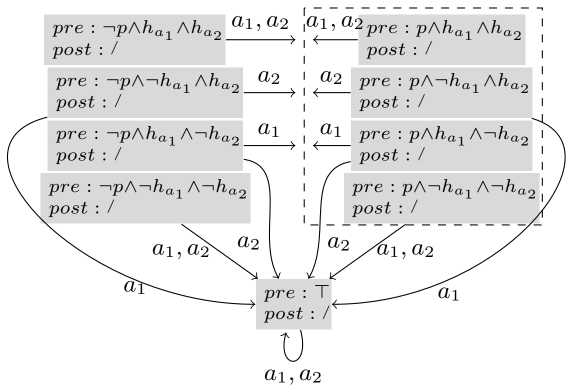
Equivalence of event models without postconditions is characterized by *action emulations* ([18]) defined as follows.

Definition 7. *Let $\mathcal{E} = (E, (\rightarrow_a^{\mathcal{E}})_{a \in Ag}, pre)$ and $\mathcal{E}' = (E', (\rightarrow_a^{\mathcal{E}'})_{a \in Ag}, pre')$ be two event models without postconditions Let Σ be the set of preconditions appearing in $\mathcal{E}$ and $\mathcal{E}'$. Let $\hat{\Sigma}$ be the set of formulas containing Σ and closed under sub-formulas and negation (see [18] for more details). Let $CS(\hat{\Sigma})$ be the set of maximal consistent subsets of $\hat{\Sigma}$. An* action emulation *AE is a set of relations $\{AE_\Gamma\}_{\Gamma \in CS(\hat{\Sigma})} \subseteq E \times E'$ such that whenever $eAE_\Gamma e'$:*

- *Invariance: $pre(e) \in \Gamma$ and $pre(e') \in \Gamma$;*
- *Zig: For all $f \in E$ and $\Gamma' \in CS(\hat{\Sigma})$, if $e \rightarrow_a^{\mathcal{E}} f$, $pre(f) \in \Gamma'$ and the formula $\left(\bigwedge_{\psi \in \Gamma} \psi \wedge \hat{K}_a \bigwedge_{\psi' \in \Gamma'} \psi'\right)$ is consistent then there exists $f' \in E'$ such that $e' \rightarrow_a^{\mathcal{E}'} f'$ and $f AE_{\Gamma'} f'$;*
- *Zag: symmetric of Zig for E'.*

Action emulation is similar to bisimulation in the sense that the types of rules are the same, except that we ask of preconditions to be in a same maximal consistent subset of $\hat{\Sigma}$. Action emulation characterizes equivalence for event models without postconditions:

Proposition 1. *$\mathcal{E}$ and $\mathcal{E}'$ are equivalent iff there is an action emulation AE between $\mathcal{E}$ and $\mathcal{E}'$ such that for all and $\Gamma \in CS(\hat{\Sigma})$ such that for all $e \in \mathcal{E}$ such that $pre(e) \in \Gamma$, there exists $e' \in \mathcal{E}'$ such that $eAE_\Gamma e'$ (and* vice versa*).*



$pre : \neg p \wedge h_{a_1} \wedge h_{a_2}$ $post : /$
$pre : \neg p \wedge \neg h_{a_1} \wedge h_{a_2}$ $post : /$
$pre : \neg p \wedge h_{a_1} \wedge \neg h_{a_2}$ $post : /$
$pre : \neg p \wedge \neg h_{a_1} \wedge \neg h_{a_2}$ $post : /$
$pre : p \wedge h_{a_1} \wedge h_{a_2}$ $post : /$
$pre : p \wedge \neg h_{a_1} \wedge h_{a_2}$ $post : /$
$pre : p \wedge h_{a_1} \wedge \neg h_{a_2}$ $post : /$
$pre : p \wedge \neg h_{a_1} \wedge \neg h_{a_2}$ $post : /$
$pre : \top$ $post : /$
a_1, a_2
a_2
a_1

Figure 4: The event model $\mathcal{A}_p$ corresponding to an attention-based announcement p for two agents a_1 and a_2. The six edges pointing to the dashed box point to all four events in the box.

2.9 Model checking

The model checking problem takes as input a pointed Kripke model M, w, a formula φ of $\mathscr{L}_{DEL}$ and returns yes if $M, w \models \varphi$; no otherwise. Sets AP and Ag are implicitly part of the input: the number of atomic propositions and the number of agents is unbounded and specified by M, w, φ.

Theorem 1. *([1]) The model checking problem is PSPACE-complete*[2].

3. SUCCINCT DEL

In classical DEL Kripke models and event models are represented explicitly. It faces some practical limits when sizes of models are exponential in the number of atomic propositions or the number of agents (see Examples 2 and 4). In this section, we provide a symbolic succinct representation for both Kripke and event models. It relies on *accessibility programs*, presented in Subsection 3.1, written in a PDL-dialect called Dynamic logic of propositional assignments (DLPA).

In Subsections 3.2 and 3.3, we define respectively succinct Kripke models and succinct event models. For both cases, we give the *semantics* (how Kripke/event models are computed from their succinct representations), the *expressiveness* (all standard DEL models can be represented in our succinct version) and the *succinctness* (some succinct models are strictly more succinct than standard DEL). In next Subsections, we present a succinct representation of the product update and the language of succinct DEL.

3.1 Language of accessibility programs

An *accessibility program*, simply called *program*, succinctly describes a relation between valuations. We write $\mathtt{u} \xrightarrow{\pi} \mathtt{v}$ for "$\mathtt{v}$ is a successor of $\mathtt{u}$ by π". The syntax for accessibility programs is defined by the following BNF.

$$\pi \quad ::= \ p{\leftarrow}\beta \mid \beta? \mid (\pi;\pi) \mid (\pi \cup \pi) \mid (\pi \cap \pi) \mid \pi^{-1}$$

where $p \in AP$, β is a Boolean formula. Program $p{\leftarrow}\beta$ is read "assign atomic proposition p to the truth value of β". Program $\beta?$ is read "test β". Program $\pi_1;\pi_2$ is read "execute π_1 then π_2". Program $(\pi_1 \cup \pi_2)$ is read "either execute π_1 or π_2". Program $(\pi_1 \cap \pi_2)$ is the intersection of π_1 and π_2. Program π^{-1} is the inverse of π. We write $set(p_1, \ldots, p_n) = (p_1{\leftarrow}\bot \cup p_1{\leftarrow}\top); \ldots; (p_n{\leftarrow}\bot \cup p_n{\leftarrow}\top)$ for the program setting arbitrary values to $p_1, \ldots, p_n$.

The semantics of accessibility programs is defined by induction as follows:

- $\mathtt{w} \xrightarrow{p \leftarrow \beta} \mathtt{u}$ iff ($\mathtt{u} = \mathtt{w} \backslash \{p\}$ and $\mathtt{w} \not\models \beta$) or ($\mathtt{u} = \mathtt{w} \cup \{p\}$ and $\mathtt{w} \models \beta$);
- $\mathtt{w} \xrightarrow{\beta?} \mathtt{u}$ iff $\mathtt{w} = \mathtt{u}$ and $\mathtt{w} \models \beta?$;
- $\mathtt{w} \xrightarrow{\pi_1;\pi_2} \mathtt{u}$ iff there exists $\mathtt{v}$ s. th. $\mathtt{w} \xrightarrow{\pi_1} \mathtt{v}$ and $\mathtt{v} \xrightarrow{\pi_2} \mathtt{u}$;
- $\mathtt{w} \xrightarrow{\pi_1 \cup \pi_2} \mathtt{u}$ iff $\mathtt{w} \xrightarrow{\pi_1} \mathtt{u}$ or $\mathtt{w} \xrightarrow{\pi_2} \mathtt{u}$;
- $\mathtt{w} \xrightarrow{\pi_1 \cap \pi_2} \mathtt{u}$ iff $\mathtt{w} \xrightarrow{\pi_1} \mathtt{u}$ and $\mathtt{w} \xrightarrow{\pi_2} \mathtt{u}$;
- $\mathtt{w} \xrightarrow{\pi^{-1}} \mathtt{u}$ iff $\mathtt{u} \xrightarrow{\pi} \mathtt{w}$.

The size of an accessibility program corresponds to the number of operators needed to write the program. For instance, the accessibility program $(p \leftarrow \top) \cup (q?; p \leftarrow \bot)$ has size 10. The models are succinctly described by means of accessibility programs. We are interested here in describing Kripke models, event models and the product update.

3.2 Succinct Kripke models

We first define the succinct representation of Kripke models, then show how to extract a Kripke model from a succinct one and *vice versa* and finally we give an example where the succinct representation is indeed strictly more succinct.

3.2.1 Definition

Definition 8. *A succinct Kripke model is a tuple $\mathfrak{M} = \langle AP_M, \beta_M, (\pi_a)_{a \in Ag} \rangle$ where AP_M is a finite set of atomic propositions, β_M is a Boolean formula over AP_M, and π_a is a program over AP_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 $\mathfrak{M}, \mathtt{w}$ with $\mathfrak{M} = \langle AP_M, \beta_M, (\pi_a)_{a \in Ag} \rangle$ is a succinct Kripke model and $\mathtt{w}$ is a valuation satisfying β_M.

3.2.2 From succinct Kripke models to Kripke models

We define the explicit Kripke model $\hat{M}(\mathfrak{M})$ associated to the succinct Kripke model $\mathfrak{M}$: the set of worlds is the set of valuations satisfying β_M and the epistemic relation $\rightarrow_a$ is the relation described by π_a.

Definition 9. *Given a succinct Kripke model $\mathfrak{M} = \langle AP_M, \beta_M, (\pi_a)_{a \in Ag} \rangle$, the Kripke model represented by $\mathfrak{M}$, noted $\hat{M}(\mathfrak{M})$ is the model $M = (W, (\rightarrow_a)_{a \in Ag}, V)$ where:*

- $W = \{\mathtt{w} \in \mathcal{V}(AP_M) \mid \mathtt{w} \models \beta_M\}$;
- $\rightarrow_a = \left\{(\mathtt{w}, \mathtt{u}) \in W^2 \mid \mathtt{w} \xrightarrow{\pi_a} \mathtt{u}\right\}$;
- $V(\mathtt{w}) = \mathtt{w}$.

[2]Actually, hardness was proven in [1] for a language with non-deterministic choice $\cup$ in the language. For a proof without such a constraint, see [6].

3.2.3 *From Kripke models to succinct models*

We define a succinct Kripke model $\mathfrak{M}_M$ representing the Kripke model M with respect to a set of propositions AP.

Definition 10. *Let $M = (W, (\rightarrow_a)_{a\in Ag}, V)$ be a Kripke model. We define the* succinct Kripke model *$\mathfrak{M}_M = \langle AP_M, \beta_M, (\pi_a)_{a\in Ag}\rangle$ where:*

- *$AP_M = AP \cup \{p_w \mid w \in W\}$;*
- *$\beta_M = \exists!(\{p_w \mid w \in W\}) \wedge \bigwedge_{w\in W} p_w \rightarrow desc(V(w))$;*
- *$\pi_a = \bigcup_{w\rightarrow_a u} p_w?; set(AP_M); p_u?$.*

The intended meaning of the fresh atomic propositions p_w is to designate the world w (as *nominals* in hybrid logic [5]). Formula β_M describes the set W and the valuation V. Program π_a performs a non-deterministic choice over edges $w \rightarrow_a u$ and then simulate the transition $w \rightarrow_a u$. The following proposition states that $\mathfrak{M}_M$ indeed represents M.

Proposition 2. *$(\hat{M}(\mathfrak{M}_M), \{p_w\} \cup V(w))$ and (M, w) are AP-bisimilar.*

Proof. We note $M = (W, (\rightarrow_a)_{a\in Ag}, V)$ and $\hat{M}(\mathfrak{M}_M) = (W', (\rightarrow'_a)_{a\in Ag}, V')$. We define $B := \{(u, p_u \cup V(u)) \mid u \in W\}$. Let us prove that B is a AP-bisimulation. Invariance of B is routine. For Zig, consider $w \in W$. For all $u \in W$, if $w \rightarrow_a u$ then we can verify that $\hat{M}(\mathfrak{M}_M), u \cup V(u) \models \beta_M$ and that $p_w \cup V(w) \xrightarrow{\pi_a} p_u \cup V(u)$ so we have $w \rightarrow'_a u$. The Zag property is symmetrical. $\square$

Example 7. *The model M from Figure 1 is modeled by the succinct Kripke model $\mathfrak{M}_M = \langle AP_M, \beta_M, (\pi_a)_{a\in Ag}\rangle$ with $AP_M = \{p, p_w, p_u\}$, $\beta_M = \exists!(\{p_u, p_w\}) \wedge (p_w \rightarrow p) \wedge (p_u \rightarrow \neg p)$ and $\pi_a = \bigcup_{v_1, v_2 \in W} p_{v_1}?; set(AP_M); p_{v_2}?$.*

3.2.4 *Succinctness of succinct Kripke models*

To show the succinctness of succinct Kripke models, we describe a family of Kripke models having exponentially more compact representations with succinct Kripke models.

Let us consider the family of models M given in Example 2 for all number of agents n. The Kripke model M is succinctly represented by the succinct Kripke model $\mathfrak{M} = \langle AP_M, \beta_M, (\pi_a)_{a\in Ag}\rangle$ defined by $\beta_M = \top$, $AP_M = \{h_a, m_a \mid a \in Ag\}$, $\pi_a = set(\{m_a\} \cup \{h_b \mid b \neq a\})$.

Formula $\beta_M = \top$ means that the set of possible worlds is the set of *all* valuations. Program π_a changes propositions agent a is uncertain of (m_a and h_b for all $b \neq a$) while the truth values of other propositions remain unchanged. Pointed Kripke models $M, \mathtt{w}$ and $\hat{M}(\mathfrak{M}), \mathtt{w}$ are bisimilar. The number of worlds in M is 2^{2n} while the size of $\mathfrak{M}$ is $O(n^2)$ (each program π_a is of size $O(n)$).

Furthermore, there is no bisimilar Kripke model M' with less worlds than M since all valuations appear once in M.

3.3 Succinct event models

We adopt a similar method for succinct event models.

3.3.1 *Definition*

Definition 11. *A succinct event model is a tuple $\mathfrak{E} = \langle AP_M, AP_E, \chi_E, (\pi_{a,E})_{a\in Ag}, \mathsf{post}\rangle$ where:*

- *AP_M and AP_E are two finite disjoint sets of atomic propositions;*
- *χ_E is a formula of $\mathscr{L}_{EL}$ over $AP_M \cup AP_E$ where atomic propositions of AP_E are not under the scope of a modal operator K_a;*
- *$\pi_{a,E}$ is a program over $AP_M \cup AP_E$ for all $a \in Ag$;*
- *post is a program over $AP_M \cup AP_E$.*

AP_M is the set of propositions used to describe preconditions while AP_E are new fresh propositions to potentially encode distinct events with the same precondition. The formula χ_E describes the set of events and their preconditions. Each $\pi_{a,E}$ corresponds to the symbolic representation of an accessibility relation $\rightarrow^{\mathcal{E}}_a$ for an agent a. post encodes the postcondition function.

3.3.2 *From succinct event models to event models*

Let $sub(\chi_E)$ be the union of AP_M and the set of subformulas of χ_E of the form $K_a\psi$ not under the scope of another $K_{a'}$ operator.

Example 8. *If $AP_M = \{p, q\}$ and $AP_E = \{p_e, p_f\}$ then $sub((\neg p_f \wedge K_a\neg K_b p) \vee K_a q) = \{p, q, K_a\neg K_b p, K_a q\}$.*

Definition 12. *Given a succinct event model $\mathfrak{E} = \langle AP_M, AP_E, \chi_E, (\pi_{a,E})_{a\in Ag}, \mathsf{post}\rangle$, the event model represented by $\mathfrak{E}$, noted $\hat{\mathcal{E}}(\mathfrak{E})$ is the model $(E, (\rightarrow^{\mathcal{E}}_a)_{a\in Ag}, pre, post)$ where*

- *$E = \{(\mathtt{v}_{pre}, \mathtt{v}_{post}) \in \mathcal{V}(sub(\chi_E) \cup AP_E) \times \mathcal{V}(AP_M)$
$\mathtt{v}_{pre} \models \chi_E$ and $\mathtt{v}_{pre|AP_M \cup AP_E} \xrightarrow{\mathsf{post}} \mathtt{v}_{post} \cup (\mathtt{v}_{pre|AP_E})\}$;*
- *$\rightarrow^{\mathcal{E}}_a = \left\{ \begin{array}{l} ((\mathtt{v}_{pre}, \mathtt{v}_{post}), (\mathtt{v}_{pre}', \mathtt{v}_{post}')) \in E^2 \\ \mid \mathtt{v}_{pre} \xrightarrow{\pi_{a,E}; set(sub(\chi_E))} \mathtt{v}_{pre}' \end{array} \right\}$;*
- *$pre((\mathtt{v}_{pre}, \mathtt{v}_{post})) = desc(\mathtt{v}_{pre|sub(\chi_E)})$;*
- *$post((\mathtt{v}_{pre}, \mathtt{v}_{post}), p) = \begin{cases} \top & \text{if } p \in \mathtt{v}_{post} \\ \bot & \text{otherwise.} \end{cases}$*

In an event $e = (\mathtt{v}_{\mathrm{pre}}, \mathtt{v}_{\mathrm{post}})$, $\mathtt{v}_{\mathrm{pre}}$ represents the valuation before the execution of e (it encodes the precondition and the truth value of propositions in AP_E). $\mathtt{v}_{\mathrm{post}}$ represents the valuation after the execution of e (it takes into account the effect of the postconditions). The relation $\rightarrow^{\mathcal{E}}_a$ is defined with the program $\pi_{a,E}$ but we also change arbitrarily propositions of $sub(\chi_E)$. The precondition of an event $e = (\mathtt{v}_{\mathrm{pre}}, \mathtt{v}_{\mathrm{post}})$ is the description of $\mathtt{v}_{\mathrm{pre}}$ restricted to $sub(\chi_E)$. For instance, if $\mathtt{v}_{\mathrm{pre}} = \{p, p_e, K_a r\}$ then $pre((\mathtt{v}_{\mathrm{pre}}, \mathtt{v}_{\mathrm{post}})) = p \wedge K_a r$. The postcondition is inferred from $\mathtt{v}_{\mathrm{post}}$.

3.3.3 *From event models to succinct event models*

We define a succinct event model $\mathfrak{E}_{\mathcal{E}}$ representing the event model $\mathcal{E}$.

Definition 13. *Let $\mathcal{E} = (E, (\rightarrow^{\mathcal{E}}_a)_{a\in Ag}, pre, post)$ be an event model. We define the succinct event model $\mathfrak{E}_{\mathcal{E}} = \langle AP_M, AP_E, \chi_E, (\pi_{a,E})_{a\in Ag}, \mathsf{post}\rangle$ where*

- *AP_M is a superset of propositions appearing in pre;*
- *$AP_E = \{p_e \mid e \in E\}$;*
- *$\chi_E = \exists!(AP_E) \wedge \bigwedge_{e\in E}(p_e \rightarrow pre(e))$;*
- *$\pi_{a,E} = \bigcup_{e\rightarrow^{\mathcal{E}}_a f} p_e?; set(AP_M \cup AP_E); p_f?$;*
- *$\mathsf{post} = \bigcup_{e\in E} p_e?; \left(\bigcap_{p\in AP_M} p \leftarrow post(e, p)\right)$.*

The fresh atomic proposition p_e designates event e. Formula χ_E describes the set E and the precondition *pre*. Program $\pi_{a,E}$ performs a non-deterministic choice in the same spirit of π_a in Definition 10. Program post non-deterministically chooses the current event e and applies the postcondition assignments in parallel. The following proposition states that $\mathfrak{E}_\mathcal{E}$ indeed represents $\mathcal{E}$.

Proposition 3. *$\hat{\mathcal{E}}(\mathfrak{E}_\mathcal{E})$ and $\mathcal{E}$ are equivalent.*

Proof. Let $\mathcal{E} = (E, (\rightarrow_a^\mathcal{E})_{a\in Ag}, pre, post)$ and $\hat{\mathcal{E}}(\mathfrak{E}_\mathcal{E}) = (E', (\rightarrow_a^{\mathcal{E}'})_{a\in Ag}, pre', post')$. Let $M = (W, (\rightarrow_a^M)_{a\in Ag}, V)$ be a Kripke model. Let B be the set of tuples $((w, e), (w, e'))$ with $w \in W$, $e \in E$ and $e' = (\mathtt{v}_{\text{pre}}{}^e, \mathtt{v}_{\text{post}}{}^e) \in E'$ such that $\mathtt{v}_{\text{pre}}{}^e = \{p_e\} \cup \mathtt{v}_e$ with $\mathtt{v}_e \subseteq sub(\chi_E), V(w) \subseteq \mathtt{v}_e, \mathtt{v}_e \models pre(e)$ and $\mathtt{v}_{\text{post}}{}^e = V(w, e)$. W. Such an e' is well defined by Definitions 12 and 13. We prove that B is a bisimulation. We have $V'((w, e')) = V((w, e))$ by Definition 12 so invariance holds. For Zig, if $(w, e) \rightarrow_a^{M\otimes\mathcal{E}} (u, f)$ then $M, u \models pre(f)$. Therefore there exists $\mathtt{v}_f \subseteq sub(\chi_E)$ such that $V(u) \subseteq \mathtt{v}_f$, $\mathtt{v}_f \models pre(f)$ and $\{p_f\} \cup \mathtt{v}_f \models \chi_E$. By definition of post, we have $V(u) \cup \{p_f\} \xrightarrow{\mathsf{post}} V((u, f)) \cup \{p_f\}$. We deduce that $f' = (\{p_f\} \cup \mathtt{v}_f, V((u, f))) \in E'$. We have $w \rightarrow_a^M u$ so $\{p_e\} \cup \mathtt{v}_e \xrightarrow{\pi_{a,E}; set(sub(\chi_E))} \{p_f\} \cup \mathtt{v}_f$. We conclude that $(u, f) \rightarrow_a^{\mathcal{E}'} (u, f')$. The Zag property is proven similarly. $\square$

Example 9. *The event model $\mathcal{E}$ of Figure 2 is modeled by the succinct event model $\mathfrak{E}_\mathcal{E} = \langle AP_M, AP_E, \chi_E, (\pi_{a,E})_{a\in Ag}, \mathsf{post}\rangle$ with $AP_M = \{p, p_w, p_u\}, AP_E = \{p_e, p_f\}$, $\chi_E = \exists!(AP_E) \wedge (p_e \rightarrow p) \wedge (p_f \rightarrow \top)$, $\pi_{a,E} = \bigcup_{e_1 \rightarrow_a^\mathcal{E} e_2} p_{e_1}?; set(AP_M \cup AP_E); p_{e_2}?$ and $\mathsf{post} = (p_e?; p \leftarrow \bot) \cup (p_f?)$ (trivial postcondition counterpart in post has been omitted).*

3.3.4 Succinctness of succinct event models

As for succinct Kripke models, we provide a family of event models having exponentially more compact representations with succinct event models.

Let us consider the family of models $(\mathcal{E}_n)_{n\in\mathbb{N}}$ given in Example 4 for all number of agents n. The event model $\mathcal{E}$ is succinctly represented by the succinct event model $\mathfrak{E} = \langle AP_M, AP_E, \chi_E, (\pi_{a,E})_{a\in Ag}, \mathsf{post}\rangle$ defined by

- $\chi_E = \top$;
- $\pi_{a,E} = (\neg p_{idle}?; (h_a?; p{\leftarrow}\top; set(\{h_b, b \in Ag\})) \cup$
  $(\neg h_a?; set(AP_M); p_{idle}{\leftarrow}\top)) \cup (p_{idle}?; set(AP_M))$;
- $\mathsf{post} = \top?$.

Atomic proposition p_{idle} intuitively means that the event *idle* is occurring (at the bottom in Figure 4). Formula $\chi_E = \top$ means that the set of possible events is unconstrained. Program $\pi_{a,E}$ works as follows: if p_{idle} is false, if h_a is true, assign $\top$ to p and arbitrarily change h_b for all $b \neq a$; if h_a is false, change valuations of propositions in AP_M and set p_{idle} to true; otherwise if p_{idle} is true, change all truth values of propositions in AP_M. The number of worlds in $\mathcal{E}$ is $2^{n+1} + 1$ while the size of $\mathfrak{E}$ is $O(n^2)$ (each program $\pi_{a,E}$ is of size $O(n)$).

Now we prove that standard event models cannot represent $\mathcal{E}_n$ as succinctly as succinct event models.

Theorem 2. *There is no propositional event model $\mathcal{E}'_n$ equivalent to $\mathcal{E}_n$ with less that 2^n events.*

Proof. By contradiction, we use the characterization of Proposition 1. We suppose that there $\mathcal{E}'_n$ has less than 2^n events, and that there is an action emulation AE between $\mathcal{E}_n$ and $\mathcal{E}'_n$. Let Σ be the set of preconditions of $\mathcal{E}_n$ and $\mathcal{E}'_n$. Note that $\hat{\Sigma}$ (defined as in Definition 7) is a set of propositional formulas. $\mathcal{E}'_n$ has less than 2^n events, so there exists $\mathtt{e}_1, \mathtt{e}_2 \in \mathcal{V}(\{p\} \cup \{h_a \mid a \in Ag\})$ with $\mathtt{e}_1 \models p$ and $\mathtt{e}_2 \models p$ and $\mathtt{e}_1 \neq \mathtt{e}_2$, and there exists an event e' of $\mathcal{E}'_n$, Γ_1, Γ_2 such that $\mathtt{e}_1 AE_{\Gamma_1} e'$ and $\mathtt{e}_2 AE_{\Gamma_2} e'$. As $\mathtt{e}_1 \neq \mathtt{e}_2$, there is an agent a such that $\mathtt{e}_1 \models \neg h_a$ and $\mathtt{e}_2 \models h_a$ (we swap $\mathtt{e}_1$ and $\mathtt{e}_2$ if $\mathtt{e}_1 \models h_a$ and $\mathtt{e}_2 \models \neg h_a$). Then $\mathtt{e}_1 \rightarrow_a^{\mathcal{E}_n} idle$. We consider the maximal consistent subset $\Gamma' = \{\varphi \in \hat{\Sigma} | \{h_a, a \in Ag\} \models \varphi\}$. We have $pre(idle) \in \Gamma'$ and the formula $\left(\bigwedge_{\psi\in\Gamma_1} \psi \wedge \hat{K}_a \bigwedge_{\psi'\in\Gamma'} \psi'\right)$ is consistent (because Γ_1 and Γ' are propositional). By Zig, there exists $f' \in E'$ such that $e' \rightarrow_a^{\mathcal{E}'_n} f'$ with $idle AE_{\Gamma'} f'$. By Zag, as $\mathtt{e}_2 AE_{\Gamma_2} e'$ and the formula $\left(\bigwedge_{\psi\in\Gamma_2} \psi \wedge \hat{K}_a \bigwedge_{\psi'\in\Gamma'} \psi'\right)$ is consistent, there exists $\mathtt{f} \in E$ such that $\mathtt{e} \rightarrow_a^{\mathcal{E}_n} \mathtt{f}$ and $\mathtt{f} AE_{\Gamma'} f'$. By invariance we obtain $pre(\mathtt{f}) \in \Gamma'$. However $pre(\mathtt{f}) = desc(\mathtt{f})$ and $p \in \mathtt{f}$ so $\{h_a, a \in Ag\} \not\models pre(\mathtt{f})$. We derive a contradiction, so $\mathcal{E}'_n$ has at least 2^n events. $\square$

3.4 Succinct product updates

Now, we generalize Definition 9 to obtain a succinct representation for updates.

Definition 14. *Let $\mathfrak{M} = \langle AP_M, \beta_M, (\pi_a)_{a\in Ag}\rangle$ be a succinct Kripke model. Let $\mathfrak{E}_1, \ldots, \mathfrak{E}_n$ be a sequence of succinct event models with $\mathfrak{E}_i = \langle AP_M, AP_{E_i}, \chi_{E_i}, (\pi_{a,E_i})_{a\in Ag}, \mathsf{post}_i\rangle$ with $AP_{E_1}, \ldots, AP_{E_n}$ being disjoint sets. The succinct product update of $\mathfrak{M}$ and $\mathfrak{E}_1, \ldots, \mathfrak{E}_n$, noted $\mathfrak{M} \otimes \mathfrak{E}_1 \otimes \cdots \otimes \mathfrak{E}_n$, is $\mathfrak{M}_n = \langle AP_n, L_n, (\pi_a^n)_{a\in Ag}\rangle$ defined by induction on n:*

- *$\mathfrak{M}_0 = \langle AP_M, [\beta_M], (\pi_a)_{a\in Ag}\rangle$;*
- *For $n \geq 1$,*
  $\mathfrak{M}_n = \langle AP_{n-1} \cup AP_{E_n}, L_{n-1} :: [\chi_{E_n}; \mathsf{post}_n], (\pi_a^n)_{a\in Ag}\rangle$ with $\pi_a^n = \mathsf{post}_n^{-1}; ((\pi_a^{n-1}; set(AP_{\mathfrak{E}_n})) \cap \pi_{a,E_n}); \mathsf{post}_n$ and $::$ is the concatenation operator.

Contrary to Definition 9, we now represent the set of worlds by a list L_n. For $n = 0$, Definition 14 and Definition 9 coincides in the sense that $[\beta_M]$ is considered the same as β_M. For $n \geq 1$, we push $\chi_{E_n}; \mathsf{post}_n$ at the end of L_{n-1}. The intuition behind the definition of accessibility programs π_a^n is as follows: we undo the effect of the postconditions of $\mathfrak{E}_n$; we then simulate the conjunction "$w \rightarrow_a w'$ and $e \rightarrow_a^\mathcal{E} e'$" of Definition 3 by executing the intersection of program $\pi_a^{n-1}; set(AP_{\mathfrak{E}_n})$ and π_{a,E_n}; finally we reapply the postconditions of $\mathfrak{E}_n$.

Next definition explains how to build the Kripke model that corresponds to the succinct product update.

Definition 15. *Given a succinct Kripke model $\mathfrak{M} = \langle AP_M, \beta_M, (\pi_a)_{a\in Ag}\rangle$ and succinct event models $\mathfrak{E}_1, \ldots, \mathfrak{E}_n$ where $\mathfrak{E}_i = \langle AP_M, AP_{E_i}, \chi_{E_i}, (\pi_{a,E_i})_{a\in Ag}, \mathsf{post}_i\rangle$, the Kripke model represented by the product update of $\mathfrak{M}$ and $\mathfrak{E}_1, ..., \mathfrak{E}_n$, noted $\hat{M}(\mathfrak{M} \otimes \mathfrak{E}_1 \otimes \cdots \otimes \mathfrak{E}_n)$ is the model $M_n = (W_n, (\rightarrow_{a,n})_{a\in Ag}, V_n)$, defined by induction on n:*

- *$M_0 = \hat{M}(\mathfrak{M})$;*
- *For $n \geq 1$: $M_n = (W_n, (\rightarrow_{a,n})_{a\in Ag}, V_n)$ where*

– $W_n = \left\{\begin{matrix}\mathtt{w} \in \mathcal{V}(AP_M \cup \bigcup_{i=1}^{n} AP_{E_i}), \textit{ s.t. there} \\ \textit{exists } \mathtt{v} \in W_{n-1} \textit{ and } \mathtt{e} \in \mathcal{V}(AP_{E_n}) \textit{ s.t.} \\ M_{n-1}, \mathtt{v} \cup \mathtt{e} \models \chi_{E_n} \textit{ and } \mathtt{v} \cup \mathtt{e} \xrightarrow{\mathsf{post}_n} \mathtt{w};\end{matrix}\right\}$

– $\rightarrow_{a,n} = \left\{(\mathtt{w}, \mathtt{u}) \in W_n^2 \mid \mathtt{w} \xrightarrow{\pi_a^n} \mathtt{u}\right\}$;

– $V_n(\mathtt{w}) = \mathtt{w}$.

We say that $\mathtt{w}$ is a state of $\mathfrak{M} \otimes \mathfrak{E}_1 \otimes \cdots \otimes \mathfrak{E}_n$ if $\mathtt{w} \in W_n$ where W_n is given in Definition 15.

PROPOSITION 4. *Let $\mathfrak{M}$ be a succinct Kripke model and $\mathfrak{E}_1, \ldots, \mathfrak{E}_n$ a sequence of succinct event models. The models $\hat{M}(\mathfrak{M}) \otimes \hat{\mathcal{E}}(\mathfrak{E}_1) \otimes \ldots \otimes \hat{\mathcal{E}}(\mathfrak{E}_n)$ and $\hat{M}(\mathfrak{M} \otimes \mathfrak{E}_1 \otimes \ldots \otimes \mathfrak{E}_n)$ are $AP_M \cup \bigcup_{i=1}^{n} AP_{E_i}$-bisimilar.*

PROOF. The result is proven by recurrence on n. Case $n = 0$ is direct. For the case $n > 1$, with the induction hypothesis it suffices to prove that $\hat{M}(\mathfrak{M} \otimes \mathfrak{E}_1 \otimes \ldots \otimes \mathfrak{E}_n)$ and $\hat{M}(\mathfrak{M} \otimes \mathfrak{E}_1 \otimes \ldots \otimes \mathfrak{E}_{n-1}) \otimes \hat{\mathcal{E}}(\mathfrak{E}_n)$ are $AP_M \cup \bigcup_{i=1}^{n} AP_{E_i}$-bisimilar. The proof technique is similar to the proof of Proposition 3, the details are left to the reader. □

When the context is clear, we write $\mathfrak{M} \otimes \vec{\mathfrak{E}}$ for $\mathfrak{M} \otimes \mathfrak{E}_1 \otimes \ldots \otimes \mathfrak{E}_n$.

3.5 Language of succinct DEL

We define the language $\mathscr{L}_{succDEL}$:

$$\varphi \ ::= \ \top \mid p \mid \neg\varphi \mid (\varphi \vee \varphi) \mid K_a\varphi \mid \langle \mathfrak{E}, \beta_0 \rangle \varphi$$

where $\mathfrak{E}$ is a succinct event model over (AP_M, AP_E) and β_0 is a Boolean formula over $AP_M \cup AP_E$.

The syntax of succinct DEL is similar to the syntax of DEL itself except that operators $\langle \mathcal{E}, E_0 \rangle$ (where $\mathcal{E}, E_0$ is a multi-pointed event model) are replaced by $\langle \mathfrak{E}, \beta_0 \rangle$ where β_0 is a Boolean formula that succinctly represents a set of events E_0. The semantics of $\langle \mathfrak{E}, \beta_0 \rangle \varphi$ is: $M, w \models \langle \mathfrak{E}, \beta_0 \rangle \varphi$ iff there exists $e \in \hat{\mathcal{E}}(\mathfrak{E})$ such that $e \models \beta_0$, $M, w \models pre(e)$ and $M \otimes \hat{\mathcal{E}}(\mathfrak{E}), (w, e) \models \varphi$ where $pre(e)$ is the precondition of e in $\hat{\mathcal{E}}(\mathfrak{E})$.

We write $\mathfrak{M} \otimes \vec{\mathfrak{E}}, \mathtt{w} \models \varphi$ for $\hat{M}(\mathfrak{M} \otimes \vec{\mathfrak{E}}), \mathtt{w} \models \varphi$.

4. MODEL CHECKING

The succinct model checking problem takes as input a pointed succinct Kripke model $\mathfrak{M}, \mathtt{w}$ and a formula φ of $\mathscr{L}_{succDEL}$, and returns yes if $\mathfrak{M}, \mathtt{w} \models \varphi$, no otherwise.

As the model checking of DLPA is PSPACE-hard [2][3], the succinct model checking problem is PSPACE-hard. Now we provide the upper bound:

THEOREM 3. *The succinct model checking problem is in PSPACE.*

To prove Theorem 3, as APTIME = PSPACE [8] (where APTIME is to the class of problems decided by an alternating Turing machine in polynomial time), we provide an alternating algorithm deciding the succinct model checking problem in polynomial time.

4.1 Background on alternating algorithms

The internal nodes of the computation tree of our algorithm on an input $\mathfrak{M}, \mathtt{w}, \varphi$ are either existential configurations (as for non-deterministic algorithms) or universal configurations. Player **(∃)** (respectively **(∀)**) chooses the next configuration in existential (universal) configurations. Leafs are either accepting or rejecting configurations. Player **(∃)** wins if the game reaches an accepting configuration.

The algorithm accepts its input $\mathfrak{M}, \mathtt{w}, \varphi$ if player **(∃)** has a winning strategy. The running time of an alternating algorithm is the height of the computation tree.

4.2 Description of our algorithm

The full pseudo-code of our algorithm is given in Figure 5 and extends algorithms for variants of DLPA given in [9] and [11] in order to handle succinct event models. The main procedure *main* for model checking succinct DEL first calls $mc_{yes}(\mathfrak{M}, w, \varphi)$. If this call is not rejecting the input (that is if $\mathfrak{M}, w \models \varphi$) then it accepts its input $\mathfrak{M}, w, \varphi$.

We implicitly define the dual versions mc_{no}, $stateof_{no}$, $issucc_{no}$ obtained from procedures mc_{yes}, $stateof_{yes}$, $issucc_{yes}$ by switching **(∃)** with **(∀)**, **and** with **or** , and indices *yes* with indices *no*. The specifications of all procedures are given in the following Theorem:

THEOREM 4. *For any succinct product update $\mathfrak{M} \otimes \vec{\mathfrak{E}}$, any valuations $\mathtt{w}, \mathtt{u}$ any formula φ of $\mathscr{L}_{succDEL}$ and any accessibility program π:*

- *mc_{yes} rejects $\mathfrak{M} \otimes \vec{\mathfrak{E}}, w, \varphi$ iff $\mathfrak{M} \otimes \vec{\mathfrak{E}}, \mathtt{w} \not\models \varphi$;*
- *mc_{no} rejects $\mathfrak{M} \otimes \vec{\mathfrak{E}}, \mathtt{w}, \varphi$ iff $\mathfrak{M} \otimes \vec{\mathfrak{E}}, \mathtt{w} \models \varphi$;*
- *$stateof_{yes}$ rejects $\mathtt{w}, \mathfrak{M} \otimes \vec{\mathfrak{E}}$ iff $\mathtt{w}$ is not a state of $\mathfrak{M} \otimes \vec{\mathfrak{E}}$;*
- *$stateof_{no}$ rejects $\mathtt{w}, \mathfrak{M} \otimes \vec{\mathfrak{E}}$ iff $\mathtt{w}$ is a state of $\mathfrak{M} \otimes \vec{\mathfrak{E}}$;*
- *$issucc_{yes}$ rejects $\mathtt{w}, \mathtt{u}, \pi$ iff $\mathtt{w} \overset{\pi}{\not\rightarrow} \mathtt{u}$;*
- *$issucc_{no}$ rejects $\mathtt{w}, \mathtt{u}, \pi$ iff $\mathtt{w} \xrightarrow{\pi} \mathtt{u}$.*

Procedures mc_{yes} **and** mc_{no}**.** The cases $\varphi = p$ and $\varphi = (\varphi_1 \vee \varphi_2)$ in mc_{yes} are straightforward. The case $\varphi = \neg\psi$ consists in calling the dual procedure mc_{no} with ψ. In the case $\varphi = K_a\psi$, we compute the program π_a^n from the Kripke model $(\mathfrak{M} \otimes \mathfrak{E}_1 \otimes ... \otimes \mathfrak{E}_n)$, given in Definition 14, then universally choose a valuation $\mathtt{u}$ and finally check that one of the three conditions holds:

- $\mathtt{u}$ is not in the set of states of $(\mathfrak{M} \otimes \mathfrak{E}_1 \otimes ... \otimes \mathfrak{E}_n)$ (the corresponding call to $stateof_{no}$ does not reject its input), meaning that the chosen valuation $\mathtt{u}$ is irrelevant.
- $\mathtt{w} \overset{\pi_a}{\not\rightarrow} \mathtt{u}$ (the call $issucc_{no}$ does not reject $\mathtt{w}, \mathtt{u}, \pi_a$), meaning that the chosen valuation $\mathtt{u}$ is irrelevant.
- or that $(\mathfrak{M} \otimes \mathfrak{E}_1 \otimes ... \otimes \mathfrak{E}_n), \mathtt{u} \models \psi$ (the call mc_{yes} does not reject $\mathfrak{M} \otimes \mathfrak{E}_1 \otimes ... \otimes \mathfrak{E}_n, u, \psi)$). This condition is particularly relevant when $\mathtt{u}$ is actually in the set of states of $(\mathfrak{M} \otimes \mathfrak{E}_1 \otimes ... \otimes \mathfrak{E}_n)$ and $\mathtt{w} \xrightarrow{\pi_a} \mathtt{u}$.

In the case $\varphi = \langle \mathfrak{E}, \beta_0 \rangle \psi$, player **(∃)** chooses a valuation $\mathtt{e}$ with $\mathtt{e} \models \beta_0$. Then the algorithm checks that $\mathtt{w} \cup \mathtt{e}$ is a state of $\mathfrak{M} \otimes \mathfrak{E}_1 \otimes ... \otimes \mathfrak{E}_n$ and that $\mathfrak{M} \otimes \mathfrak{E}_1 \otimes ... \otimes \mathfrak{E}_n, \mathfrak{E}, (\mathtt{w} \cup \mathtt{e}) \models \psi$.

Procedures $stateof_{yes}$ **and** $stateof_{no}$**.** The call $stateof_{yes}(w, \mathfrak{M}, \mathfrak{E}_1, ..., \mathfrak{E}_n)$ rejects if and only if the valuation u does not correspond to a state of $\mathfrak{M}, \mathfrak{E}_1, ..., \mathfrak{E}_n$. In order to check that the valuation $\mathtt{u}$ is not a state of $\mathfrak{M} \otimes \mathfrak{E}_1 \otimes ... \otimes \mathfrak{E}_n$, the procedure mimics Definition 14 and distinguishes two cases. In the case $n = 0$, we check that β_M holds in $\mathtt{w}$. In the case $n \geq 1$, we first check χ_{E_n}. Then we simulate the program post_n by universally choosing a valuation $\mathtt{u}$ such that $\mathtt{u} \xrightarrow{\mathsf{post}_n} \mathtt{w}$. Finally, we check that χ_{E_n} holds in $\mathtt{u}$ (we check that the precondition of $\mathfrak{E}_n$ holds).

Procedures $issucc_{yes}$ and $issucc_{no}$. The cases over π follow the semantics for accessibility programs (Subsection 3.1).

```
Procedure $main(\mathfrak{M}, \mathtt{w}, \varphi)$
  $mc_{yes}(\mathfrak{M}, \mathtt{w}, \varphi)$
  **accept**
```

```
Procedure $mc_{yes}(\mathfrak{M} \otimes \mathfrak{E}_1 \otimes ... \otimes \mathfrak{E}_n, \mathtt{w}, \varphi)$
  Case $\varphi = p$: if $\mathtt{w} \not\models p$ then reject
  Case $\varphi = (\varphi_1 \vee \varphi_2)$:
    $(\exists)$ choose $i \in \{1, 2\}$
    $mc_{yes}((\mathfrak{M} \otimes \mathfrak{E}_1 \otimes ... \otimes \mathfrak{E}_n), \mathtt{w}, \varphi_i)$
  Case $\varphi = \neg\psi$: $mc_{no}((\mathfrak{M} \otimes \mathfrak{E}_1 \otimes ... \otimes \mathfrak{E}_n), \mathtt{w}, \psi)$.
  Case $\varphi = K_a\psi$:
    Get $\pi_a^n$ from $(\mathfrak{M} \otimes \mathfrak{E}_1 \otimes ... \otimes \mathfrak{E}_n)$.
    $(\forall)$ choose $\mathtt{u}$ over $AP_{\mathfrak{M} \otimes \mathfrak{E}_1 \otimes ... \otimes \mathfrak{E}_n}$
    $(\exists) stateof_{no}(\mathtt{u}, (\mathfrak{M} \otimes \mathfrak{E}_1 \otimes ... \otimes \mathfrak{E}_n))$
    or $issucc_{no}(\mathtt{w}, \mathtt{u}, \pi_a)$ or $mc_{yes}((\mathfrak{M} \otimes \mathfrak{E}_1 \otimes ... \otimes \mathfrak{E}_n), \mathtt{u}, \psi)$
  Case $\varphi = \langle \mathfrak{E}, \beta_0 \rangle \psi$:
    $(\exists)$ choose $e$ such that $\mathtt{e} \models \beta_0$
    $(\forall) stateof_{yes}(\mathtt{w} \cup \mathtt{e}, (\mathfrak{M} \otimes \mathfrak{E}_1 \otimes ... \otimes \mathfrak{E}_n \otimes \mathfrak{E}_n))$
    and $mc_{yes}(\mathfrak{M} \otimes \mathfrak{E}_1 \otimes ... \otimes \mathfrak{E}_n \otimes \mathfrak{E}, (\mathtt{w} \cup \mathtt{e}), \psi)$.
```

```
Procedure $stateof_{yes}(\mathtt{w}, \mathfrak{M} \otimes \mathfrak{E}_1 \otimes ... \otimes \mathfrak{E}_n)$
  Case $n = 0$: $mc_{yes}(\mathfrak{M}, \mathtt{w}, \beta_M)$
  Case $n > 0$:
    $(\exists)$ choose $\mathtt{u} \in \mathcal{V}(AP_{\mathfrak{M} \otimes \mathfrak{E}_1 \otimes ... \otimes \mathfrak{E}_n})$
    $(\forall) issucc_{yes}(\mathtt{u}, \mathtt{w}, \mathsf{post}_n)$
    and $stateof_{yes}(\mathtt{u}_{|AP_{\mathfrak{M} \otimes \mathfrak{E}_1 ... \otimes \mathfrak{E}_{n-1}}}, \mathfrak{M} \otimes \mathfrak{E}_1, ..., \otimes \mathfrak{E}_{n-1})$
    and $mc_{yes}(\mathfrak{M}, \mathfrak{E}_1, ..., \mathfrak{E}_{n-1}, \mathtt{u}_{|AP_{\mathfrak{M} \otimes \mathfrak{E}_1 \otimes ... \otimes \mathfrak{E}_{n-1}}},, \chi_{E_n})$
```

```
Procedure $issucc_{yes}(\mathtt{w}, \mathtt{u}, \pi)$
  Case $\pi = p \leftarrow \beta$: if $(\mathtt{w} \models \beta$ and $u \neq \mathtt{w} \cup \{p\})$
                          or $(\mathtt{w} \not\models \beta$ and $\mathtt{u} \neq \mathtt{w} \backslash \{p\})$ then reject
  Case $\pi = \beta?$: if $\mathtt{w} \neq \mathtt{u}$ or $\mathtt{w} \not\models \beta$ then reject
  Case $\pi = (\pi_1; \pi_2)$:
    $(\exists)$ choose a valuation $\mathtt{v}$ $(\forall) issucc_{yes}(\mathtt{w}, \mathtt{v}, \pi_1)$
    and $issucc_{yes}(\mathtt{v}, \mathtt{u}, \pi_2)$
  Case $\pi = (\pi_1 \cup \pi_2)$:
    $(\exists)$ choose $i \in \{1, 2\}$
    $issucc_{yes}(\mathtt{w}, \mathtt{u}, \pi_i)$
  Case $\pi = (\pi_1 \cap \pi_2)$:
    $(\forall)$ choose $i \in \{1, 2\}$
    $issucc_{yes}(\mathtt{w}, \mathtt{u}, \pi_i)$
  Case $\pi = \pi'^{-1}$: $issucc_{yes}(\mathtt{u}, \mathtt{w}, \pi')$.
```

Figure 5: Pseudo-code

4.3 Sketch of proofs

LEMMA 1. *The procedure main runs in polynomial time in the size of* $(\mathfrak{M} \otimes \mathfrak{E}_1 \otimes ... \otimes \mathfrak{E}_n, \mathtt{w}, \varphi)$.

PROOF. The size of the argument of the procedures is strictly decreasing at each step, so the execution is in linear time in respect to the size of the input. Thus, Mc runs in polynomial time in respect to the size of $(\mathfrak{M} \otimes \mathfrak{E}_1 \otimes ... \otimes \mathfrak{E}_n, \mathtt{w}, \varphi)$. Therefore, the height of the computation tree is polynomial in the size of the input. $\square$

Theorem 4 is proved by a mutual induction for mc_{no}, $stateof_{no}$, $issucc_{no}$ obtained from mc_{yes}, $stateof_{yes}$, $issucc_{yes}$ on the size of their inputs.

4.4 Impact

The translation, with straightforward base cases,

$$tr(\langle \mathcal{E}, E_0 \rangle \varphi) = \langle \mathfrak{E}_{\mathcal{E}}, \bigvee_{e \in E_0} p_e \wedge \bigwedge_{e \notin E_0} \neg p_e \rangle tr(\varphi) \qquad (*)$$

is such that $M, w \models \varphi$ iff $\mathfrak{M}_M, p_w \wedge desc(V(w)) \models tr(\varphi)$. Thus, Theorem 3 gives an alternative proof for the PSPACE upper bound of the model checking of DEL.

Interestingly, if for some $\mathcal{E}, E_0$ whose size depends on the input (as the attention-based announcement in Example 4) and for which there exists a succinct pointed event model $\mathfrak{E}, \beta_0$ of polynomial size in the input, then, instead of (*), we use:

$$tr(\langle \mathcal{E}, E_0 \rangle \varphi) = \langle \mathfrak{E}, \beta_0 \rangle tr(\varphi)$$

and the model checking remains in PSPACE. For instance, the model checking of an epistemic formula that contains an arbitrary finite number of agents and attention-based announcement modalities $[p!]_{at}$ is in PSPACE.

5. CONCLUSION AND PERSPECTIVES

In this paper, we provide an exponentially more succinct version of Dynamic epistemic logic (DEL) whose model checking problem remains in PSPACE. We conjecture that a model checking procedure for our succinct language may use BDD techniques of [14].

It opens a long-term research track for studying algorithmic problems in DEL such as satisfiability problem and epistemic planning in their succinct versions.

The succinct satisfiability problem is the following decision problem: determine whether a formula $\varphi \in \mathscr{L}_{succDEL}$ is satisfiable. There exists a tableau method for determining whether a formula $\varphi \in \mathscr{L}_{DEL}$ is satisfiable [1] that yields a non-deterministic algorithm in exponential time for the satisfiability problem. We conjecture that the tableau method can be adapted for a formula $\varphi \in \mathscr{L}_{succDEL}$ in order to prove that the succinct satisfiability problem is in NEXPTIME.

Charrier et al. [10] provides a succinct language for iterations of event models: they provide modalities $\langle (\mathcal{E}, e)^i \rangle$ where i is an integer written in binary instead of the explicit sequence $(\mathcal{E}, e); \ldots; (\mathcal{E}, e)$. The corresponding model checking of DEL is PSPACE-complete even with such succinct iterations, when preconditions are propositional and with trivial postconditions. It would be interesting to study the extension of our succinct DEL with succinct iterations.

Also, interestingly, the satisfiability problem of attention-based announcement logic is PSPACE-complete [7] even if corresponding event models are exponential in the number of agents (see Figure 4). So the gap between PSPACE and NEXPTIME is not due to the size of models but in the very structure of event models. We claim that our succinct version DEL may help in characterizing fragments of DEL with lower complexity for model checking/satisfiability problem.

Acknowledgements.

We would like to thank Malvin Gattinger and Sophie Pinchinat for their useful insights and the anonymous reviewers for their comments. We also acknowledge support from CNRS Défi INFIniTi - AAP 2017.

REFERENCES

[1] G. Aucher 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, January 7-9, 2013*, 2013.

[2] P. Balbiani, 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.

[3] P. Balbiani, A. Herzig, and N. Troquard. Dynamic logic of propositional assignments: A well-behaved variant of PDL. In *28th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2013, New Orleans, LA, USA, June 25-28, 2013*, pages 143–152, 2013.

[4] A. Baltag, L. S. Moss, and S. Solecki. The logic of public announcements, common knowledge, and private suspicions. In *Proceedings of the 7th conference on Theoretical aspects of rationality and knowledge*, pages 43–56. Morgan Kaufmann Publishers Inc., 1998.

[5] P. Blackburn. Representation, reasoning, and relational structures: a hybrid logic manifesto. *Logic Journal of the IGPL*, 8(3):339–365, 2000.

[6] T. Bolander, 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, July 25-31, 2015*, pages 2791–2797, 2015.

[7] T. Bolander, H. van Ditmarsch, A. Herzig, E. Lorini, P. Pardo, and F. Schwarzentruber. Announcements to attentive agents. *Journal of Logic, Language and Information*, 25(1):1–35, 2016.

[8] A. K. Chandra and L. J. Stockmeyer. Alternation. In *Proc. of FOCS'76*, pages 98–108, 1976.

[9] T. Charrier, A. Herzig, E. Lorini, F. Schwarzentruber, and F. Maffre. Building epistemic logic from observations and public announcements. In *International Conference on Principles of Knowledge Representation and Reasoning (KR), CapeTown. AAAI Press*, 2016.

[10] T. Charrier, B. Maubert, and F. Schwarzentruber. On the impact of modal depth in epistemic planning. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016*, pages 1030–1036, 2016.

[11] T. Charrier 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, May 4-8, 2015*, pages 1471–1479, 2015.

[12] T. Charrier and F. Schwarzentruber. A succinct language for dynamic epistemic logic (long version). Research report, IRISA, 2017.

[13] C. H. Papadimitriou. *Computational complexity.* John Wiley and Sons Ltd., 2003.

[14] J. Van Benthem, J. van Eijck, M. Gattinger, and K. Su. Symbolic model checking for dynamic epistemic logic. In *Logic, Rationality, and Interaction*, pages 366–378. Springer, 2015.

[15] H. van Ditmarsch and B. Kooi. One hundred prisoners and a light bulb. In *One Hundred Prisoners and a Light Bulb*, pages 83–94. Springer, 2015.

[16] H. van Ditmarsch, W. van der Hoek, and B. Kooi. *Dynamic Epistemic Logic*. Springer, Dordecht, 2008.

[17] H. P. van Ditmarsch and B. P. Kooi. Semantic results for ontic and epistemic change. *CoRR*, abs/cs/0610093, 2006.

[18] J. van Eijck, J. Ruan, and T. Sadzik. Action emulation. *Synthese*, 185(Supplement-1):131–151, 2012.