A Symbolic Representation for Probabilistic Dynamic Epistemic Logic

Sébastien Gamblin
Normandie Univ, UNICAEN, ENSICAEN, CNRS, GREYC
14000 Caen, France
sebastien.gamblin@unicaen.fr

Alexandre Niveau
Normandie Univ, UNICAEN, ENSICAEN, CNRS, GREYC
14000 Caen, France
alexandre.niveau@unicaen.fr

Maroua Bouzid
Normandie Univ, UNICAEN, ENSICAEN, CNRS, GREYC
14000 Caen, France
maroua.bouzid-mouaddib@unicaen.fr

ABSTRACT

Probabilistic Dynamic Epistemic Logic (PDEL) is a formalism for reasoning about the higher-order probabilistic knowledge of agents, and about how this knowledge changes when events occur. While PDEL has been studied for its theoretical appeal, it was only ever applied to toy examples: the combinatorial explosion of probabilistic Kripke structures makes the PDEL framework impractical for realistic applications, such as card games.

This paper is a first step towards the use of PDEL in more practical settings: in line with recent work applying ideas from symbolic model checking to (non-probabilistic) DEL, we propose a "symbolic" representation of probabilistic Kripke structures as pseudo-Boolean functions, which can be represented with several data structures of the decision diagram family, in particular Algebraic Decision Diagrams (ADDs). We show that ADDs scale much better than explicit Kripke structures, and that they allow for efficient symbolic model checking, even on the realistic example of the Hanabi card game, thus paving the way towards the practical application of epistemic planning techniques.

KEYWORDS

Knowledge Representation; Probabilistic Dynamic Epistemic Logic; Symbolic Model Checking; Hanabi; Decision Diagrams

ACM Reference Format:
Sébastien Gamblin, Alexandre Niveau, and Maroua Bouzid. 2022. A Symbolic Representation for Probabilistic Dynamic Epistemic Logic. In *Proc. of the 21st International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2022), Online, May 9–13, 2022*, IFAAMAS, 9 pages.

1 INTRODUCTION

The card game Hanabi [3] has recently drawn interest from the AI community [2]. Hanabi is a multiplayer cooperative game with imperfect information in which *higher-order knowledge* plays a very important role, i.e., players need to make decisions depending on what they know about what other players know, and so on. With the ultimate goal of computing strategies for games like Hanabi, our focus is on approaches based on Dynamic Epistemic Logic (DEL) [26], a formalism allowing one to reason about the higher-order knowledge of agents, and about how this knowledge changes when events occur. DEL constructs allow for an elegant approach to multi-agent epistemic planning [4], and more recently, to the problems of controller and distributed strategy synthesis in adversarial games [18].

However, in many realistic games, in particular those with imperfect or incomplete information such as Hanabi, winning strategies generally do not exist; what is usually desired in these cases is an *optimal* strategy, one that maximizes the expectation of a victory. A natural direction is to study an adaptation of epistemic planning to Probabilistic Dynamic Epistemic Logic (PDEL) [10, 17, 23], a generalization of DEL which is interpreted on Kripke structures augmented with probabilistic information. Yet, while there has been effort lately to make DEL useable in practice by applying ideas from symbolic model checking, thus avoiding the combinatorial explosion of explicit Kripke structures [7, 8, 15, 24], there has been no such work for PDEL. To the best of our knowledge, it remains an entirely theoretical framework, only ever applied to toy examples.

This paper is a first step towards the use of PDEL in more practical settings: we propose a "symbolic" representation of probabilistic Kripke structures as pseudo-Boolean functions, which can be represented with several data structures of the decision diagram family. Our experiments using Algebraic Decision Diagrams (ADDs) on a modelization of Hanabi show that the size of these representations scale much better than explicit Kripke structures, while allowing for efficient symbolic model checking, even for near-realistic game sizes. The next step is to generalize epistemic planning to a probabilistic setting, so as to study optimal strategy synthesis for games with imperfect or incomplete information; our results indicate that such a generalization would not only be of theoretical interest, but could also be useful in practice.

After presenting some background, such as Hanabi and PDEL (§ 2), we introduce our symbolic representation and model checking procedure (§ 3), then report about experimental results (§ 4).

2 BACKGROUND

2.1 Hanabi

Hanabi [3] is a cooperative card game where players must play their cards in order; its specificity is that players can never look at their cards, although they see those of other players and can give them information. There are five card *colors*, and five card *values* (numbers from 1 to 5) per color. The goal is to *play* as many cards as possible on the table, but the cards of each color must be played in increasing order (an error costs one *red token*, out of three in total). At their turn, each player can either play a card on the table, make an *announcement* (which costs one *blue token*), or *discard* a card (which gives one blue token back). An announcement consists in

Proc. of the 21st International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2022), P. Faliszewski, V. Mascardi, C. Pelachaud, M.E. Taylor (eds.), May 9–13, 2022, Online.

showing to another player the positions of all cards of one specific color or value in their hand.

Hanabi is an interesting game to study because finding efficient strategies seems to require the modeling of a *theory of mind* – this is how Google/DeepMind researchers explain that they found this game to be harder to solve than Go [2]. Epistemic logic approaches being specifically designed for reasoning about higher-order knowledge, they look like a good fit to the "Hanabi challenge". Now, non-probabilistic DEL is enough to represent Hanabi, and as it happens a recent study coupled such a DEL representation with a Monte-Carlo Tree Search algorithm [20]. Yet, we would like to study exact approaches to optimal strategy synthesis; but in Hanabi it is not always possible to make a safe move, so looking for "strong plans" in an epistemic planning version of Hanabi is doomed to fail. Probabilistic reasoning is necessary to evaluate strategies according to the *expectation* of rewards. Concretely, we want to be able to ask questions such as "how likely is it that Alice knows that this card is green?" or "is it more likely than not that playing this card would cost us a red token?", which are very natural to express in PDEL.

We will represent the "physical" state of the game (independant from the knowledge acquired by the players) using propositional variables, which basically describe where each card is located. Clearly, this is not sufficient to represent the real state of a game: the knowledge of each player about their cards evolves during a game, and in order to reason about how best to play, it is necessary to have a model of this knowledge – we use *epistemic logic* for this.

2.2 Probabilistic Epistemic Logic

We fix a finite set $\mathfrak{A}$ of *agents* and a countable set *PS* of propositional symbols. We first define probabilistic Kripke structures, which model the epistemic and probabilistic knowledge of agents. There are various options for the latter; e.g., some general definitions rely on σ-algebras [11]. Although our approach can be applied to richer settings, in this paper we use the simplest case of probability functions. For reasons explained later, we do not require our probability functions to be normalized; we borrow the term "lottery" from van Eijck and Schwarzentruber [27] (with a slightly different definition) to refer to non-normalized probability functions.

Definition 1 (X-lottery). *An X-*lottery *is a function $L : X \to \mathbb{Q}^+$ from a (finite) set X of outcomes to the set of positive rationals. We call* support *of L the set of outcomes to which L assigns a nonzero value:* $\mathrm{supp}(L) := \{x \in X \mid L(x) > 0\}$. *Last, we define* NL, *the* normalization *of L, to be the X-lottery associating to each $x \in X$ the value $\mathrm{NL}(x) := L(x)/\sum_{x' \in X} L(x')$ if $\mathrm{supp}(L) \neq \emptyset$, else 0.*

Definition 2 ((probabilistic) Kripke structure). *A* (probabilistic) Kripke structure *for a vocabulary $V \subseteq PS$ is a tuple $\mathcal{M} = \langle W, R, \mu, \mathrm{val}\rangle$ such that: (i) W is a nonempty finite set of* worlds, *(ii) R associates to each agent $\mathfrak{a} \in \mathfrak{A}$ an* accessibility relation $R_\mathfrak{a} \subseteq W \times W$, *(iii) μ associates to each agent $\mathfrak{a} \in \mathfrak{A}$ and each world $w \in W$ a W-lottery $\mu_\mathfrak{a}(w)$, and (iv)* val *is a* valuation function $W \to 2^V$, *indicating the set of propositions that are true in each world. A* pointed Kripke structure *is a pair, denoted $\mathcal{M}_w$, of a Kripke structure and one of its worlds, which is interpreted as being the "real" world.*

This definition is that of van Benthem et al. [23], except that μ is defined using W-lotteries rather than probability functions. As long as the semantics of the logic language is adapted accordingly with an additional normalization (as we do later in Def. 5), this is completely equivalent, since we only consider finite sets of worlds. Note also that although we took the "lottery" terminology from van Eijck and Schwarzentruber [27], we do not use their simplified setting, but the full-blown PDEL of van Benthem et al. [23]. Finally, even though we formally define valuations as sets of true atoms, in examples we use a more explicit, *assignment*-like notation: for example, valuation $\{q\} \in 2^{\{p,q,r\}}$ is written $\bar{p}q\bar{r}$.

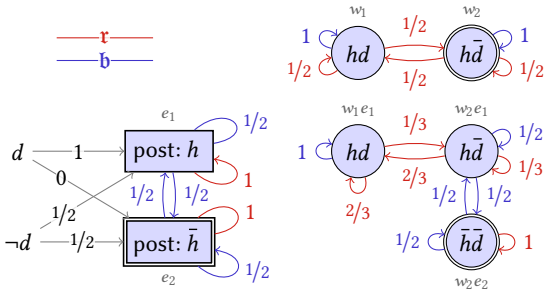


Figure 1: A Kripke structure (top; Ex. 3), an update model (left; Ex. 7), and their product update (bottom right; Ex. 10).

Intuitively, the accessibility relation of agent $\mathfrak{a}$ links worlds that $\mathfrak{a}$ considers as *undistinguishable*, and $\mu_a(w_1)(w_2)$ gives the probability that $\mathfrak{a}$ assigns to being in world w_2 when it is actually in w_1.

Example 3. *Consider a heads-up coin on the table: h (for "heads") is true and everyone knows it. The coin is either double-headed (d) or it is fair (¬d): agent $\mathfrak{b}$ (for "blue") knows the value of d, but not agent $\mathfrak{r}$ (for "red"), who considers that the probability of the coin being fair is 1/2. The corresponding Kripke structure is illustrated in Figure 1 (top right), and formally defined as follows: $W = \{w_1, w_2\}$, $R_\mathfrak{r} = \{\langle w_1, w_1\rangle, \langle w_1, w_2\rangle, \langle w_2, w_2\rangle, \langle w_2, w_1\rangle\}$, $R_\mathfrak{b} = \{\langle w_1, w_1\rangle, \langle w_2, w_2\rangle\}$, $\mathrm{val}(w_1) = hd$, $\mathrm{val}(w_2) = h\bar{d}$, $\mu_\mathfrak{r}(w_1) = \mu_\mathfrak{r}(w_2) = \{w_1 : 1/2, w_2 : 1/2\}$, $\mu_\mathfrak{b}(w_1) = \{w_1 : 1\}$, $\mu_\mathfrak{b}(w_2) = \{w_2 : 1\}$. Note that in all our examples, the support of the lotteries will always coincide with the accessibility relation, so we do not draw the latter; but in all generality, lotteries are not restricted to accessible worlds.*

Having both an accessibility relation and probability functions may seem redundant, but it has two advantages: (i) it allows one to model an unquantified uncertainty between possible worlds, which is not the same as assigning them a uniform probability, and (ii) it allows one to distinguish between having a 0 probability of being true and being truly impossible. See Fagin and Halpern [11] for an in-depth discussion. Yet, it is quite natural and intuitive to consider only what Demey and Kooi [10] call *i-consistent* structures, in which $\mu_\mathfrak{a}(w)$ gives a positive probability only to states accessible via R. All examples in this paper indeed respect this constraint, although our framework does not enforce it (nor any other constraint between R and μ), because it is actually simpler this way. Any useful constraint can be easily added anyway, if need be.

Similarly, we make no hypothesis about accessibility relations – contrary to a large part of the DEL literature, which generally focuses on the modal logic S5, and thus on *equivalence relations* for R. Although for simplicity our examples are all S5, our approach

does not rely on this condition, and can be directly applied to more expressive logics such as KD45.

Definition 4 (probabilistic epistemic language $\mathcal{L}_{\text{PEL}}$). *Given a vocabulary V, the language $\mathcal{L}_{\text{PEL}}(V)$ of probabilistic epistemic logic is defined by the following grammar in Backus-Naur form:*

$$\phi ::= p \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid \Box_{\mathfrak{a}}\phi \mid \alpha_1 \text{Pr}_{\mathfrak{a}}(\phi_1) + \cdots + \alpha_k \text{Pr}_{\mathfrak{a}}(\phi_k) \geq \beta$$

where $p \in V$, $\mathfrak{a} \in \mathfrak{A}$, $\alpha_1, \ldots, \alpha_k, \beta$ are rationals, and $k \geq 1$.

We also use parentheses in formulas for disambiguation, as well as usual abbreviations like $\vee$, $\rightarrow$, $\leftrightarrow$, and $\sum_{i=1}^{k} \alpha_i \text{Pr}_{\mathfrak{a}}(\phi_i)$. In addition to propositional formulas (the language of which we denote $\mathcal{L}_{\text{prop}}(V)$), $\mathcal{L}_{\text{PEL}}(V)$ allows for higher-order formulas such as $\Box_{\mathfrak{b}}(\text{Pr}_{\mathfrak{r}}(d) \geq 1/2)$ or $\text{Pr}_{\mathfrak{b}}(\Box_{\mathfrak{r}} d) \geq 1/2$. For example, the first formula states that agent $\mathfrak{b}$ knows that agent $\mathfrak{r}$ estimates that the probability that d holds is greater than or equal to $1/2$. Next, we explain how one decides whether a formula in $\mathcal{L}_{\text{PEL}}$ holds in a given state of epistemic and probabilistic knowledge:

Definition 5 (semantics of PEL). *Let $\mathcal{M}_w$ be a pointed Kripke structure for a vocabulary V. We define the semantics of PEL inductively as follows, where $p \in V$, $k \geq 1$, $\phi, \psi, \phi_1, \ldots, \phi_k \in \mathcal{L}_{\text{PEL}}$, $\mathfrak{a} \in \mathfrak{A}$, and $\alpha_1, \ldots, \alpha_k, \beta \in \mathbb{Q}$:*

$\mathcal{M}_w \models p$ *iff* $p \in \text{val}(w)$
$\mathcal{M}_w \models \neg\phi$ *iff* $\mathcal{M}_w \not\models \phi$
$\mathcal{M}_w \models \phi \wedge \psi$ *iff* $\mathcal{M}_w \models \phi$ *and* $\mathcal{M}_w \models \psi$
$\mathcal{M}_w \models \Box_{\mathfrak{a}}\phi$ *iff* $\mathcal{M}_{w'} \models \phi$ *for all w' s.t.* $\langle w, w' \rangle \in R_{\mathfrak{a}}$
$\mathcal{M}_w \models \sum_{i=1}^{k} \alpha_i \text{Pr}_{\mathfrak{a}}(\phi_i) \geq \beta$ *iff*

$$\sum_{i=1}^{k} \alpha_i \left(\sum_{w' \mid \mathcal{M}_{w'} \models \phi_i} \text{N}\mu_{\mathfrak{a}}(w)(w') \right) \geq \beta$$

Note that the last case uses *normalized* lotteries, i.e., probability functions, or the constant 0 if the lottery has empty support. The definition is thus equivalent to that of van Benthem et al. [23].

2.3 Updating Knowledge

Kripke structures, probabilistic or not, only represent a *static* state of knowledge. To account for changes, e.g. due to player actions, they have to be coupled with *update models* (also called "action models" or "event models"). They have several definitions in the probabilistic epistemic logic literature; the following is adapted from that of van Benthem et al. [23], notably adding postcondition functions and using lotteries instead of normalized probability functions.

Definition 6 ((probabilistic) update model). *A* (probabilistic) update model *for vocabulary V is a tuple $\mathcal{E} = \langle E, R^{\mathcal{E}}, \mu^{\mathcal{E}}, \Phi, \text{pre}, \text{post} \rangle$, where (i) E is a nonempty set of* events, *(ii) $R^{\mathcal{E}}$ associates to each agent $\mathfrak{a} \in \mathfrak{A}$ an* accessibility relation $R^{\mathcal{E}}_{\mathfrak{a}} \subseteq E \times E$, *(iii) $\mu^{\mathcal{E}}$ associates to each agent $\mathfrak{a} \in \mathfrak{A}$ and each event $e \in E$ an E-lottery $\mu_{\mathfrak{a}}(e)$, (iv) Φ is a finite set of pairwise inconsistent formulas from $\mathcal{L}_{\text{PDEL}}(V)$ (Def. 9),*[1] *called* preconditions, *(v)* pre *assigns to each precondition $\phi \in \Phi$ an E-lottery* $\text{pre}(\phi)$, *(vi)* post *is a* postcondition *function $E \times V \rightarrow \mathcal{L}_{\text{prop}}(V)$. A* pointed update model *is a pair, denoted $\mathcal{E}_e$, of an update model and one of its events, considered to be the actual event taking place.*

[1] For simplicity, as is customary in the DEL literature, we directly use $\mathcal{L}_{\text{PDEL}}$ (which is only defined later and depends on Def. 6) as the language of preconditions; behind the apparent cyclic definitions only lies a harmless simultaneous recursion [23].

Preconditions in this definition are different from those of classical DEL update models [26], in which there is simply one precondition formula for each event. As van Benthem et al. [23] argue, this allows one to independently consider *observation probabilities*, given by $\mu^{\mathcal{E}}$, which quantify the distinguishability of events; and *occurrence probabilities*, given by Φ and pre, which indicate the probability with which each event can occur in each situation. More precisely, given a pointed structure $\mathcal{M}_w$, the *occurence probability of e at w*, denoted by $\text{pre}(w)(e)$, is defined as $\text{pre}(\phi)(e)$, where ϕ is the formula in Φ such that $\mathcal{M}_w \models \phi$, or 0 if there is no such ϕ. We take this more general approach in this paper – except that we use lotteries instead of probability functions, which is once again completely equivalent since we have modified the semantics of the Pr operator in Def. 5.

The postcondition functions (not used by van Benthem et al. [23], but present in other probabilistic DEL approaches such as that of van Eijck and Schwarzentruber [27]) allow update models to have *ontic effects*, i.e., to change the state of the world and not only the knowledge state of agents [25]; this is indeed crucial if we want to use PDEL to reason about games such as Hanabi. By fixing $\text{post}(e, x) = x$ for all $e \in E$ and all $x \in V$, we get a purely epistemic update model, with no ontic effect.

Example 7. *Let us continue Ex. 3 by considering a coin flip: agent $\mathfrak{r}$ tosses the coin, but hides the result from agent $\mathfrak{b}$. If the coin is fair (i.e., if d is false), the two possible results are equiprobable, but if it is double-headed, the result cannot be tails. These specifications define the following update model, illustrated in Fig. 1 (left): $E = \{e_1, e_2\}$, $R^{\mathcal{E}}_{\mathfrak{r}} = \{\langle e_1, e_1 \rangle, \langle e_2, e_2 \rangle\}$, $R^{\mathcal{E}}_{\mathfrak{b}} = E \times E$, $\Phi = \{d, \neg d\}$, $\text{pre}(d) = \{e_1 : 1/2, e_2 : 1/2\}$, $\text{pre}(\neg d) = \{e_1 : 1, e_2 : 0\}$, $\mu^{\mathcal{E}}_{\mathfrak{r}}(e_1)$ associates 1 to e_1 and 0 to e_2, $\mu^{\mathcal{E}}_{\mathfrak{r}}(e_2)$ associates 1 to e_2 and 0 to e_1, $\mu^{\mathcal{E}}_{\mathfrak{b}}(e_1) = \mu^{\mathcal{E}}_{\mathfrak{b}}(e_2) = 1/2$, $\text{post}(e_1, h) = \top$, $\text{post}(e_2, h) = \bot$. Note how the agents' distinct levels of information are modeled by observation probabilities between the events themselves: $\mathfrak{b}$ always considers that the two events are equiprobable, while in each possible event $\mathfrak{r}$ gives probability 1 to the current event and 0 to the other. On the other hand, the fairness of the coin controls the probability of each event taking place: in particular, because of occurrence probabilities, if d holds, only e_1 can occur.*

Now that we have update models to represent events that can occur in the real world (either by a voluntary action of an agent, or not), and how these events are perceived by the agents, we will explain how we can use them. The mechanism used in PDEL to compute the new state after an event took place is called *product update*; it is more or less a simple Cartesian product of the Kripke structure representing the previous knowledge state and of the update model representing the event.

Definition 8 (probabilistic product update). *Let $\mathcal{M}_w$ be a pointed Kripke structure and $\mathcal{E}_e$ be a pointed update model for the same vocabulary V. The* product update *of $\mathcal{M}$ by $\mathcal{E}$ is the Kripke structure $\mathcal{M} \otimes \mathcal{E} := \langle W^{\otimes}, R^{\otimes}, \mu^{\otimes}, \text{val}^{\otimes} \rangle$, where*

- $W^{\otimes} := \{\langle w, e \rangle \in W \times E \mid \text{pre}(w)(e) > 0\}$;
- $\langle \langle w_1, e_1 \rangle, \langle w_2, e_2 \rangle \rangle \in R^{\otimes}_{\mathfrak{a}}$ *iff* $\langle w_1, w_2 \rangle \in R_{\mathfrak{a}}$ *and* $\langle e_1, e_2 \rangle \in R^{\mathcal{E}}_{\mathfrak{a}}$;
- $\mu^{\otimes}_{\mathfrak{a}}(\langle w_1, e_1 \rangle)(\langle w_2, e_2 \rangle) := \mu_{\mathfrak{a}}(w_1)(w_2) \cdot \text{pre}(w_2)(e_2) \cdot \mu^{\mathcal{E}}_{\mathfrak{a}}(e_1)(e_2)$;

- $\mathrm{val}^{\otimes}(\langle w, e\rangle) := \{p \in V \mid \mathcal{M}w \models \mathrm{post}(e, p)\}$.

The product update of the pointed structures is accordingly defined as $\mathcal{M}w \otimes \mathcal{E}_e := (\mathcal{M} \otimes \mathcal{E})_{\langle w,e\rangle}$.

Note that contrary to the original PDEL definition on which ours is based [23], no normalization is needed to compute $\mu^{\otimes}$ – the one we added in the semantics of Pr (Def. 5) is sufficient. Now, thanks to the product update, the logic language can be augmented with a new operator that means "after the application of this event, this formula is true". This is how the logic becomes "dynamic".

Definition 9 (PDEL). *Given a vocabulary V, the* probabilistic dynamic epistemic language $\mathcal{L}_{\mathrm{PDEL}}(V)$ *is defined by the BNF of* $\mathcal{L}_{\mathrm{PEL}}$ *(Def. 4) augmented with the production rule* $\phi ::= [\mathcal{E}_e]\phi$*, where* $\mathcal{E}_e$ *is a pointed probabilistic update model. The semantics of this additional operator is defined as follows:* $\mathcal{M}w \models [\mathcal{E}_e]\phi$ *iff* $\mathrm{pre}(w, e) > 0 \implies (\mathcal{M} \otimes \mathcal{E})_{\langle w,e\rangle} \models \phi$.

Example 10. *We go on with the coin flip example. Figure 1 shows the Kripke structure of the initial situation (top right), the update model (bottom left) and the resulting structure (bottom right). Double borders indicate pointed worlds and events. First, note how there is no world labeled "w_1e_2" in the final structure: indeed, the occurrence probability* $\mathrm{pre}(w_1)(e_2)$ *is* 0 *($\neg h$ cannot occur if d holds). Before the toss, we have* $\Box_{\mathfrak{r}} h \wedge \Box_{\mathfrak{b}} h \wedge \Box_{\mathfrak{b}} \neg d \wedge \neg\Box_{\mathfrak{r}} \neg d$, $\mathrm{Pr}_{\mathfrak{b}} \neg d \geq 1$, $\mathrm{Pr}_{\mathfrak{r}} \neg d \geq 1/2$ *and* $\neg(\mathrm{Pr}_{\mathfrak{r}} \neg d \geq 0.51)$. *After the toss, $\mathfrak{b}$ still knows that the coin is fair, $\mathfrak{r}$ knows that it is not heads up, and $\mathfrak{b}$ does not know that it is heads up, i.e.,* $[\mathcal{E}_{e_2}](\Box_{\mathfrak{b}} \neg d \wedge \Box_{\mathfrak{r}} \neg h \wedge \neg\Box_{\mathfrak{b}} h)$ *holds; but interestingly,* $[\mathcal{E}_{e_2}](\Box_{\mathfrak{r}} \neg d \wedge \Box_{\mathfrak{b}}(\mathrm{Pr}_{\mathfrak{r}}(\neg d) \geq 1/3))$ *also holds, i.e., after the toss $\mathfrak{r}$ knows that the coin is fair (because of w_1e_2 having been filtered out), and $\mathfrak{b}$ knows that the probability that $\mathfrak{r}$ assigns to the coin being fair is more than 1/3, even without knowing how the coin landed.*

We are interested in the *model checking* problem, which consists in deciding, given a pointed Kripke structure $\mathcal{M}w$ and a formula $\phi \in \mathcal{L}_{\mathrm{PDEL}}$, whether $\mathcal{M}w \models \phi$.

3 SYMBOLIC REPRESENTATION

Because of the combinatorial nature of Kripke structures, the size of such representations quickly becomes huge in practical examples, even moderately realistic. To avoid this problem, an idea is to use a "symbolic" representation of Kripke structures, in which redundancies are factored out, so that knowledge states remain scalable while still allowing for reasonably efficient model checking. The idea of symbolic model checking [19] has recently been applied to dynamic epistemic logic, the symbolic representation of Kripke structures being either *accessibility programs* (called "mental programs" in [7, 8]) or Binary Decision Diagrams (BDDs) [15, 24], BDDs being a well-known efficient representation of Boolean formulas [5].

We now show how these concepts apply to probabilistic DEL; contrary to Shirazi and Amir [22], who also represent probabilistic Kripke structures in a factored way (using Bayesian networks), we are not only interested in static structures. In order to represent probabilities, we must go beyond the usual Boolean formulas of symbolic approaches to DEL, and use pseudo-Boolean functions.

Definition 11 (pseudo-Boolean function). *Given $X = \{x_1, \ldots, x_n\} \subseteq PS$ a finite set of propositional symbols, we call* pseudo-Boolean function (PBF) over X *a total function of the type* $f\colon 2^X \to \mathbb{Q}$. *(The somewhat abusive expression "over X" must be understood as "over variables from X.")*

There are several ways to represent pseudo-Boolean functions, notably generalizations of BDDs; let us mention Algebraic Decision Diagrams (ADDs) [1], Semiring-Labelled Decision Diagrams (SLDDs)[13, 28], Affine Algebraic Decision Diagrams (AADDs) [21], and Probabilistic Sentential Decision Diagrams (PSDDs) [16]. These languages are of varying *succinctness* (i.e., some are able to represent PBFs more compactly than others) and do not have the same efficiency for operations (such as summing PBFs or "forgetting" variables). There is a tradeoff to be found, depending on the application; systematically studying and quantifying such tradeoffs is the goal of the literature about the *knowledge compilation map* [9, 12]. In this paper, we remain as general as possible and only talk about PBFs; it should be implicitly understood that they are represented in some efficient language, such as ADD, which we use in our experiments and briefly present in § 4.

Before going on, we need to introduce some conventions and notations. First, we use a convenient notation for PBFs in examples: e.g., given mutually inconsistent formulas $\phi, \psi \in \mathcal{L}_{\mathrm{prop}}(X)$, notation $\{\phi\colon 0.4, \psi\colon 0.8\}$ designates a PBF associating 0.4 to models of ϕ, 0.8 to models of ψ, and implicitly 0 to other valuations. We consider that Boolean functions, i.e., functions of the form $f\colon 2^X \to \mathbb{B}$, are particular PBFs, and for simplicity we often identify propositional formulas ϕ over X with the Boolean function they represent; i.e., we see ϕ as the Boolean function $\{\phi\colon 1\}$. We call support of f, written $\mathrm{supp}(f)$, the set $Y \subseteq 2^X$ where f is nonzero: $\mathrm{supp}(f) = \{v \in 2^X \mid f(v) \neq 0\}$. Note that if ϕ is a propositional formula over X, $\mathrm{supp}(\phi)$ is the model set of ϕ.

For two PBFs f and g over $X \subseteq PS$, we denote $\mathrm{Cut}_{\geq}(f, g)$ the Boolean function associating a valuation $v \in 2^X$ to 1 if $f(v) \geq g(v)$, and to 0 otherwise. Let $X, Y, Z \subseteq PS$ such that $X \subseteq Y$ and $Y \cap Z = \emptyset$, and let $m\colon X \to \mathcal{L}_{\mathrm{prop}}(Z)$. For f a PBF over Y, we denote by $[m]f$ the substitution of X via m in f, defined by $[m]f\colon v \mapsto f((v \setminus Z) \cup \{x \in X \mid (v \cap Z) \models m(x)\})$. A special case of substitution is *variable renaming*; we write $[X \triangleright X']f$ for the substitution of variables in X by variables in X' when the mapping between X and X' is clear from the context. Following Gattinger [15], given $v \in 2^X$, we denote $v \sqsubseteq X$ the formula on X that sets all variables in v to true and all the others to false, i.e., $v \sqsubseteq X := \bigwedge_{p \in v} p \wedge \bigwedge_{p \in X \setminus v} \neg p$; and we write $[v \sqsubseteq X]f$ for $[v \to \top, (X \setminus v) \to \bot]f$ (the *conditioning* of f by valuation $v \in 2^X$).

Finally, let $\odot\colon \mathbb{Q} \times \mathbb{Q} \to \mathbb{Q}$ be an associative and commutative operation with a neutral element (such as addition, multiplication, min, max, or the Boolean connectives $\vee$ and $\wedge$), let $X \subseteq Y \subseteq PS$ and let f be a PBF over Y. The $\odot$*-marginalization* of X in f, denoted $\mathrm{Marg}^{\odot}_X(f)$, is the function $g\colon 2^{Y \setminus X} \to \mathbb{Q}$ defined by $g(v) := \bigodot_{v' \in 2^Y \mid v = (v' \setminus X)} f(v')$. Note that if f is a Boolean function, $\vee$-marginalization (resp. $\wedge$-marginalization) of Y corresponds to *existentially (resp. universally) forgetting* the variables in Y. We write $\mathrm{Forget}^{\exists}_Y(f) = \mathrm{Marg}^{\vee}_Y(f)$ and $\mathrm{Forget}^{\forall}_Y(f) = \mathrm{Marg}^{\wedge}_Y(f)$.

3.1 Static Structures

Let us now show how to represent a probabilistic Kripke structure with PBFs. The basic idea is to identify worlds with their valuations, so that propositional formulas over V directly represent sets of

worlds. Obviously, it is not possible in the general case: two distinct worlds in a Kripke structure can have the same valuation. The symbolic representation only works for structures whose valuation function val is injective, i.e., such that $\forall w_1, w_2 \in W\colon w_1 \neq w_2 \rightarrow \mathsf{val}(w_1) \neq \mathsf{val}(w_2)$; we call such structures *valuation-injective*. In this case, for a valuation $v \in 2^V$ such that $\exists w \in W\colon \mathsf{val}(w) = v$, we write $\mathsf{val}^{-1}(v)$ to designate w.

While valuation-injective Kripke structures have considerably reduced expressiveness (e.g., the satisfiable formula $\Box_\mathfrak{a} p \wedge \neg\Box_\mathfrak{b}(\Box_\mathfrak{a} p \vee \Box_\mathfrak{a}\neg p)$ has no valuation-injective S5 or KD45 model for vocabulary $V = \{p\}$), the setting still applies to a lot of games – notably those in which all uncertainty is reducible to uncertainty about the "physical state" of the game. This is the case for Hanabi: e.g., even though Alice does not know what the other players know about her cards, she can enumerate the possible game states (i.e. her possible hands). For each one, assuming it is the actual game state, she knows exactly what other players know. This would not be the case if there were private announcements (e.g., one player gives information to another without being heard by the others) or secret actions (e.g., two players switch one of their cards without others noticing). Note that this restriction is not unusual; it also holds for existing approaches to symbolic model checking for epistemic logic. Moreover, if more expressiveness is needed, it is always possible to add fresh symbols to distinguish worlds with the same valuation – but this is only necessary for the *initial* Kripke structure: as we will see, symbolic updates automatically disambiguate worlds that would have the same valuation in explicit form.

We now define "symbolic" Kripke structures; they are an extension of the "belief structures" of the SMCDEL framework [15, 24].

Definition 12 (symbolic Kripke structure). *A* symbolic Kripke structure *is a tuple $\mathcal{F} = \langle V, \theta, \Omega, \Pi\rangle$ such that:*

- *$V \subseteq PS$ is a finite set of symbols called the* vocabulary*,*
- *θ is a Boolean function over V, called the* state law*;*
- *Ω associates to each agent $\mathfrak{a} \in \mathfrak{A}$ a Boolean function $\Omega_\mathfrak{a}$ over $V \cup V'$, called the* observation law*;*
- *Π associates to each agent $\mathfrak{a} \in \mathfrak{A}$ a PBF $\Pi_\mathfrak{a}$ over $V \cup V'$, called the* probability law*.*

Any $s \subseteq V$ such that $s \models \theta$ is called a state *of $\mathcal{F}$, and the pair $\langle \mathcal{F}, s\rangle$ is called a* pointed symbolic Kripke structure *and denoted $\mathcal{F}_s$.*

These symbolic Kripke structures are very generic, since, as we already mentioned, there is no constraint as to how the pseudo-Boolean functions defining the three laws are represented. The choice of concrete representations can thus depend on the intended tradeoff between spatial and temporal efficiency for various applications. Anyway, it should be clear that symbolic Kripke structure can yield exponential space savings: a trivial example (for one agent) is $F = \langle V, \top, \top, 1\rangle$ (where 1 is the constant PBF), which represents a Kripke structure with $2^{|V|}$ distinct worlds (that are undistinguishable and all considered equally probable by the agent).

We now formalize the relationship between "explicit" Kripke structures and symbolic ones by defining how to translate a Kripke structure into a symbolic one, which is rather straightforward:

Definition 13 (symbolic representation of a Kripke structure). *The* symbolic representation *of a valuation-injective Kripke structure $\mathcal{M} = \langle W, R, \mu, \mathsf{val}\rangle$ for vocabulary V is the symbolic Kripke structure* $\mathsf{symb}(\mathcal{M}) = \langle V, \theta, \Omega, \Pi\rangle$ *where (i) $\theta := \bigvee_{w\in W}(\mathsf{val}(w) \sqsubseteq V)$; (ii) $\Omega_\mathfrak{a} := \bigvee_{w_1 R_\mathfrak{a} w_2}(\mathsf{val}(w_1) \sqsubseteq V) \wedge (\mathsf{val}(w_2) \sqsubseteq V)'$; (iii) for any $v_1, v_2 \in 2^V$, $\Pi_\mathfrak{a}(v_1 \cup (v_2)') := \mu_\mathfrak{a}(\mathsf{val}^{-1}(v_1), \mathsf{val}^{-1}(v_2))$ if val^{-1} is defined for both v_1 and v_2, and 0 otherwise.*

The key point is that each state of $\mathsf{symb}(\mathcal{M})$ corresponds to a unique world of $\mathcal{M}$, thanks to valuation-injectivity; it should be clear that $\Omega_\mathfrak{a}$ and $\Pi_\mathfrak{a}$ are direct representations of $R_\mathfrak{a}$ and $\mu_\mathfrak{a}$. However, Def. 12 does not constrain Ω and Π as to the values they give to valuations in 2^V that are not states (i.e., not models of θ). This can be useful, depending on the data structures used to represent PBFs, because it can yield more compact representations (by removing dependencies between variables). Although Def. 13 does not allow this for simplicity (keeping symb a function), it would be easy to relax the notion of symbolic representation of a Kripke structure by taking advantage of this flexibility. Note also that symb allows any Kripke structure – be it S5, KD45, or another combination of epistemic logic axioms – to be represented as a symbolic Kripke structure.

Example 14. *The symbolic representation of the Kripke structure in Ex. 3 is $\mathcal{F} = \langle V, \theta, \Omega, \Pi\rangle$ with $V = \{h, d\}$, $\theta \equiv h$, $\Omega_\mathfrak{r} \equiv h \wedge h'$, $\Omega_\mathfrak{b} \equiv h \wedge h' \wedge d \leftrightarrow d'$, $\Pi_\mathfrak{r} = 1/2$, and $\Pi_\mathfrak{b} = \{\Omega_\mathfrak{b}\colon 1\}$.*

3.2 Model Checking on Static Structures

In order to decide whether a Kripke structure represented symbolically is a model of an $\mathcal{L}_{\mathrm{PEL}}$ formula, we can build a Boolean function over its vocabulary whose models are the worlds of the structure in which the formula holds. This can be done using dynamic programming thanks to the following inductive definition, also extending that of Gattinger [15], van Benthem et al. [24].

Definition 15 (local Boolean translation of formula). *Let $\mathcal{F} = \langle V, \theta, \Omega, \Pi\rangle$ be a symbolic Kripke structure and ϕ be a formula in $\mathcal{L}_{\mathrm{PEL}}(V)$. The* local Boolean translation of ϕ in $\mathcal{F}$*, denoted $\|\phi\|_\mathcal{F}$, is the Boolean function defined inductively as follows:*

- $\|p\|_\mathcal{F} := p$;
- $\|\neg\phi\|_\mathcal{F} := \neg\|\phi\|_\mathcal{F}$;
- $\|\phi \wedge \psi\|_\mathcal{F} := \|\phi\|_\mathcal{F} \wedge \|\psi\|_\mathcal{F}$;
- $\|\Box_\mathfrak{a}\phi\|_\mathcal{F} := \mathsf{Forget}^\forall_{V'}((\Omega_\mathfrak{a} \wedge \theta') \rightarrow (\|\phi\|_\mathcal{F})')$;
- $\|\sum_{i=1}^k \alpha_i \mathrm{Pr}_\mathfrak{a}(\phi_i) \geq \beta\|_\mathcal{F} := \mathsf{Cut}_{\geq}(\mathcal{N}, \beta \times \mathcal{D})$, *where $\mathcal{N}$ is defined as $\mathcal{N} := \sum_{i=1}^k \left(\alpha_i \cdot \mathsf{Marg}^+_{V'}((\Pi_\mathfrak{a} \times \theta') \times (\|\phi_i\|_\mathcal{F})')\right)$, and $\mathcal{D} := \mathsf{nonzero}(\mathsf{Marg}^+_{V'}(\Pi_\mathfrak{a}))$, in which* $\mathsf{nonzero}(f)$ *is the PBF associating v to $f(v)$ if $f(v) \neq 0$, and to ∞ otherwise.*

Let us briefly explain the last point: the first marginalization buids a PBF $\mathcal{N}$ associating to each world the sum of the lottery values of all the worlds that satisfy ϕ. The second marginalization computes the denominator in the definition of $\mathrm{N}\mu_\mathfrak{a}$ (Def. 1). The nonzero function is a trick for taking care of the empty support lotteries (it works even when $\beta = 0$ and when $\beta < 0$, provided that we fix $0 \times \infty = 0$ by convention). Finally, the Cut operator removes worlds that do not meet the threshold β. Note that the normalization used here would also be required if the probability law was normalized; it is not a consequence of using lotteries.

It should be clear that the complexity of building the local translation of a formula depends on the concrete representations used.

Nonetheless, all necessary operations can be considered as elementary operations on PBF representations; they are notably used by Fargier et al. [12] as criteria to compare the efficiency of several languages of the decision diagram family. Finally, the following result can be proved by induction:

Proposition 16 (symbolic model checking on PEL). *Let $\mathcal{M} = \langle W, R, \mu, \text{val}\rangle$ be a valuation-injective Kripke structure; for any formula $\phi \in \mathcal{L}_{\text{PEL}}$ and any $w \in W$, it holds that $\mathcal{M}_w \models \phi \iff \text{val}(w) \models \|\phi\|_{\text{symb}(\mathcal{M})}$.*

3.3 Symbolic Updates

Now that we have defined a symbolic Kripke structure and showed how PEL model checking can be done on it, we will show how these structures can be updated through symbolic update models. We would like to use the same principle as for Kripke structures to represent accessibility relations and observation probabilities of update models; however, that is not possible, since events have no valuation. Moreover, each event has to be linked to its preconditions (which can be epistemic formulas) and postconditions. We use the approach of van Benthem et al. [24] in their SMCDEL framework, that is, we label events using a fresh vocabulary V^+, effectively "pretending" that events have valuations. Of course, we chose this valuation function carefully to ensure that it be injective. The symbolic representation of accessibility relations and observation probability functions of update models can then be represented exactly as those of Kripke structures – respectively as Boolean functions and PBFs over the double vocabulary $V^+ \cup V^{+'}$.

Definition 17 (event labeling function). *Let $\mathcal{E} = \langle E, R^{\mathcal{E}}, \mu^{\mathcal{E}}, \text{pre}, \text{post}\rangle$ be a Kripke structure for vocabulary V. An* event labeling function *λ for $\mathcal{E}$ is an injective function $\lambda\colon E \to 2^{V^+}$, where $V^+ \subseteq PS$ is a set of fresh symbols ($V \cap V^+ = \emptyset$) that we denote* $\text{voc}(\lambda)$.

Given an event labeling function, we denote by $\mathcal{M}_{\mathcal{E},\lambda}$ the Kripke structure for vocabulary $\text{voc}(\lambda)$ defined as $\langle E, R^{\mathcal{E}}, \mu^{\mathcal{E}}, \lambda\rangle$; it is the "underlying" Kripke structure of $\mathcal{E}$, using the labeling function as its valuation function. We can then use the observation and probability laws of $\text{symb}(\mathcal{M}_{\mathcal{E},\lambda})$ to represent $R^{\mathcal{E}}$ and $\mu^{\mathcal{E}}$ symbolically.

Preconditions, however, are a bit more complex to handle in PDEL than in DEL. In SMCDEL, symbolic events (called "transformers") contain an event law θ^+ which is a possibly epistemic formula over $V \cup V^+$ that links each precondition formula to the set of events of which it is a precondition. In PDEL, each precondition formula does not correspond to a "flat" set of events, but to an *occurrence probability function* over events. We chose to remain as simple as possible in our symbolic update models by keeping the set Φ of precondition formulas as is, and by including for each $\phi \in \Phi$ a PBF $\theta^{\text{pre}}(\phi)$ over V^+ representing its occurrence probability function (which is, as usual in this paper, represented by a lottery).

Finally, we manage postconditions following the SMCDEL approach [15]: our symbolic update models feature a subset $V_{\_} \subseteq V$ of all propositional symbols that are modified by at least one event, together with a Boolean function $\theta_{\_}(p)$ over $V \cup V^+$ for each such symbol $p \in V_{\_}$. An assignment of $V \cup V^+$ can be interpreted as a pair $\langle w, e\rangle$ of a world and an event; the value given by $\theta_{\_}(p)$ to this assignment is the value that the postcondition function of e gives to p if the previous world was w.

Definition 18 (symbolic update model). *A* symbolic update model *for the vocabulary V is a tuple $\chi = \langle V^+, \Omega^+, \Pi^+, \Phi, \theta^{\text{pre}}, V_{\_}, \theta_{\_}\rangle$ where: (i) V^+ is a set of fresh propositional symbols: $V \cap V^+ = \emptyset$, (ii) Ω^+ associates to each agent $\mathfrak{a} \in \mathfrak{A}$ a Boolean function over $V^+ \cup V^{+'}$ called the* event observation law *of $\mathfrak{a}$, (iii) Π^+ associates to each agent $\mathfrak{a} \in \mathfrak{A}$ a PBF over $V^+ \cup V^{+'}$ called the* probabilistic observation law *of $\mathfrak{a}$, (iv) Φ is a set of pairwise inconsistent formulas of $\mathcal{L}_{\text{PDEL}}(V)$, called* precondition formulas*, (v) θ^{pre} associates to each $\phi \in \Phi$ a PBF $\theta^{\text{pre}}(\phi)$ over V^+ called its* occurrence probability law*, (vi) $V_{\_} \subseteq V$ is a subset of the original vocabulary called the* modified subset*, (vii) $\theta_{\_}$ associates to each modified symbol $p \in V_{\_}$ a Boolean function $\theta_{\_}(p)$ over $V \cup V^+$ called the* change law *of p. Any $x \subseteq V^+$ such that $\exists \phi \in \Phi\colon \theta^{\text{pre}}(\phi)(x) > 0$ is called a state of χ.*

Definition 19 (symbolic representation of an update model). *Let $\mathcal{E} = \langle E, R^{\mathcal{E}}, \mu^{\mathcal{E}}, \Phi, \text{pre}, \text{post}\rangle$ be an update model for vocabulary V and λ be an event labeling function for $\mathcal{E}$. The* symbolic representation of $\mathcal{E}$ via λ *is the symbolic update model* $\text{symb}_\lambda(\mathcal{E}) = \langle V^+, \Omega^+, \Pi^+, \Phi, \theta^{\text{pre}}, V_{\_}, \theta_{\_}\rangle$ *where*

- $V^+ \coloneqq \text{voc}(\lambda)$,
- Ω^+ *is the observation law of* $\text{symb}(\mathcal{M}_{\mathcal{E},\lambda})$,
- Π^+ *is the probability law of* $\text{symb}(\mathcal{M}_{\mathcal{E},\lambda})$,
- *for $\phi \in \Phi$ and $v \in V^+$, $\theta^{\text{pre}}(\phi)(v) \coloneqq \text{pre}(\phi)(\lambda^{-1}(v))$ if λ^{-1} is defined on v, and 0 otherwise,*
- $V_{\_} \coloneqq \{p \in V \mid \exists e \in E\colon \text{post}(e,p) \not\equiv p\}$, *and*
- $\theta_{\_}(p) \coloneqq \bigwedge_{e\in E}(\lambda(e) \leftrightarrow \text{post}(e,p))$.

Example 20. *The symbolic representation of the coin flip update model (Ex. 7) via the event labeling function defined as $\lambda(e_1) \coloneqq q$ and $\lambda(e_2) \coloneqq \bar{q}$ is $\chi = \langle V^+, \Omega^+, \Pi^+, \Phi, \theta^{\text{pre}}, V_{\_}, \theta_{\_}\rangle$ where $V^+ = \{q\}$, $\Omega^+_{\mathfrak{r}} \equiv q \leftrightarrow q'$, $\Omega^+_{\mathfrak{b}} \equiv \top$, $\Pi^+_{\mathfrak{r}} = \{\Omega^+_{\mathfrak{r}}\colon 1\}$, $\Pi^+_{\mathfrak{b}} = 0.5$, $\Phi = \{d, \neg d\}$, $\theta^{\text{pre}}(d) = \{q\colon 1\}$, $\theta^{\text{pre}}(\neg d) = 1/2$, $V_{\_} = \{h\}$, $\theta_{\_}(h) \equiv q$.*

We can now explain how to compute the product update on symbolic structures. Following the SMCDEL framework, modified symbols are *historicized*: they are replaced by fresh symbols for each application of the product update (i.e., each update has its own fresh vocabulary $V^\circ_{\_}$). Thanks to this, the resulting structure keeps track of the previous valuations. Now, an aspect that is quite different from SMCDEL is the treatment of preconditions. We do not have a θ^+ (see above) of which we can compute the local translation to filter states. We must compute the translation of each $\phi \in \Phi$ and multiply the result by the occurrence probability law $\theta^{\text{pre}}(\phi)$. The PBF we obtain represents the occurrence probability function of *each state that satisfies ϕ*. Summing the PBFs of all ϕ's yields the occurrence probability function of each state (this is sound because the ϕ's are required to be mutually inconsistent). The resulting PBF can be seen as a probabilistic equivalent of the θ^+ from SMCDEL, and we thus denote it by Θ^+. In particular, the support of Θ^+ contains all pairs state-event whose occurrence probability is nonzero, i.e., it corresponds exactly to the event law θ^+ of SMCDEL transformers.

Example 21. *We illustrate the construction of Θ^+ for the symbolic update model χ of Ex. 20: Θ^+ is defined as $\theta^{\text{pre}}(d)\times\|d\|_{\mathcal{F}} + \theta^{\text{pre}}(\neg d)\times\|\neg d\|_{\mathcal{F}}$, but since the precondition formulas are Boolean, their local translations do not depend on $\mathcal{F}$. Hence, Θ^+ only has to be computed once and can be reused for all product updates with χ. Specifically, $\Theta^+ = \{q\colon 1\}\times\{d\colon 1\} + 1/2\times\{\neg d\colon 1\}$ and thus $\Theta^+ = \{d\wedge q\colon 1, \neg d\colon 1/2\}$. We can understand Θ^+ as follows: if the coin is fair, there is a 50%*

chance that it lands heads (q) or tails ($\bar{q}$), but if it is rigged, there is a 100% chance that event q occurs (i.e., the coin must land heads). The support of Θ^+ is the model set of $\neg d \vee (d \wedge q) \equiv d \to q$, which represents the fact that the event labeled $\bar{q}$ cannot occur if d is true.

Definition 22 (symbolic product update). *Given a pointed symbolic Kripke structure $\mathcal{F}_s$ and a pointed symbolic update model χ_x for vocabulary V, the* symbolic product update of $\mathcal{F} = \langle V, \theta, \Omega, \Pi\rangle$ by $\chi = \langle V^+, \Omega^+, \Pi^+, \Phi, \theta^{\mathsf{pre}}, V_-, \theta_-\rangle$, *is the symbolic Kripke structure $\mathcal{F} \otimes \chi = \langle V^\otimes, \theta^\otimes, \Omega^\otimes, \Pi^\otimes\rangle$ defined by*

- $V^\otimes := V \cup V^+ \cup V^\circ_-$, *where V°_- is a set of fresh symbols (distinct from the V°_- of previous product updates),*
- $\theta^\otimes := [V_- \triangleright V^\circ_-](\theta \wedge \mathsf{supp}(\Theta^+)) \wedge \bigwedge_{p \in V_-}(p \leftrightarrow [V_- \triangleright V^\circ_-](\theta_-(p)))$,
- $\Omega^\otimes_a := ([V_- \triangleright V^\circ_-][(V_-)' \triangleright (V^\circ_-)']\Omega_a) \wedge \Omega^+_a$, *and*
- $\Pi^\otimes_a := ([V_- \triangleright V^\circ_-][(V_-)' \triangleright (V^\circ_-)'](\Pi_a \times (\Theta^+)')) \times \Pi^+$,

where $\Theta^+ := \sum_{\phi \in \Phi}(\theta^{\mathsf{pre}}(\phi) \times \|\phi\|_{\mathcal{F}})$. The product update of the pointed structures is $(\mathcal{F} \otimes \chi)_{s \cdot x}$, with $s \cdot x := (s \backslash V_-) \cup (s \cap V_-)^\circ \cup x \cup \{p \in V_- \mid s \cup x \models \theta_-(p)\}$.

The symbolic product update behaves exactly like the explicit one, in that the occurrence probabilities and the two observation probabilities are simply multiplied together. In particular, there is no need to normalize anything, which is one of the main advantages of using lotteries: normalization is only ever required during model checking, when the formula contains a $\sum \alpha \geq \beta$ construct. The lottery approach can thus be seen as a "lazy" one, in which dealing with probabilities is deferred until the last moment. This is interesting in terms of the number of operations to be carried out, but the decisive advantage of this lazy approach is that it can significantly shorten PBF representations (intuitively, normalizing makes them more constrained and thus generally larger) which in turn greatly improves the efficiency of operations.

Example 23. *Let us compute the product update of $\mathcal{F}$ from Ex. 14 and χ from Ex. 20. First, the new vocabulary is $V^\otimes = \{h, d, q, h^\circ\}$: we get variables h and d from $\mathcal{F}$, q is from the labeling function, and h° is the historicized version of h, that stores the value it had before the update (there is no d° because d is not modified by any event, so it is not in the modified vocabulary of χ). The new state law is $\theta^\otimes \equiv h^\circ \wedge (d \to q) \wedge (h \leftrightarrow q)$, i.e., h was true before the update, the event labelled $\bar{q}$ cannot occur if d was true, and the new value of h is directly given by the event that occurred). The observation law is given by $\Omega^\otimes_{\mathfrak{r}} = h^\circ \wedge h^{\circ\prime} \wedge q \leftrightarrow q'$, $\Omega^\otimes_{\mathfrak{b}} = h^\circ \wedge h^{\circ\prime} \wedge d \leftrightarrow d'$. Before showing the probability law, let us note that valuations $h^\circ qhd$, $h^\circ qh\bar{d}$, $h^\circ \bar{q}h\bar{d}$ respectively represent worlds w_1e_1, w_2e_1, w_2e_2 in Fig. 1 (bottom right). For example, $h^\circ \bar{q}h\bar{d}$ must be read like this: h was true before and is now false, the event that occurred is $\bar{q}$, and d was and is still false. Denoting $\phi_{w_1e_1}$ (resp. $\phi_{w_2e_1}$, $\phi_{w_2e_2}$) the term corresponding to valuation $h^\circ qhd$ (resp. $h^\circ qh\bar{d}$, $h^\circ \bar{q}h\bar{d}$), we have $\Pi_{\mathfrak{r}} = \{(\phi_{w_1e_1} \vee \phi_{w_2e_1}) \wedge \phi'_{w_1e_1} : 2/3, (\phi_{w_1e_1} \vee \phi_{w_2e_1}) \wedge \phi'_{w_2e_1} : 1/3, \phi_{w_2e_2} \wedge \phi'_{w_2e_2} : 1\}$ and $\Pi_{\mathfrak{b}} = \{\phi_{w_1e_1} \wedge \phi'_{w_1e_1} : 1, (\phi_{w_2e_1} \vee \phi_{w_2e_2}) \wedge (\phi_{w_2e_1} \vee \phi_{w_2e_2})' : 1/2\}$.*

The following result ties everything together:

Proposition 24. *Let V be a vocabulary; let $\mathcal{M}_w$ be a pointed Kripke structure for V; let $\mathcal{E}_e$ be a pointed update model for V and λ an event labeling function for $\mathcal{E}$; and let $\phi \in \mathcal{L}_{\mathrm{PEL}}(V)$. We have*

$$(\mathcal{M} \otimes \mathcal{E})_{\langle w,e\rangle} \models \phi \iff \mathsf{val}(w) \cdot \lambda(e) \models \|\phi\|_{\mathsf{symb}(\mathcal{M}) \otimes \mathsf{symb}_\lambda(\mathcal{E})}$$

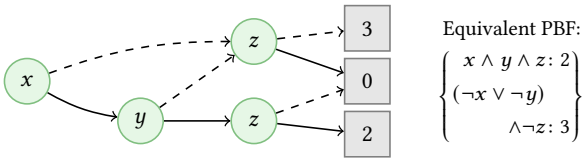


Figure 2: Example of an ADD (dashed arcs represent "if variable is false" and solid ones, "if variable is true").

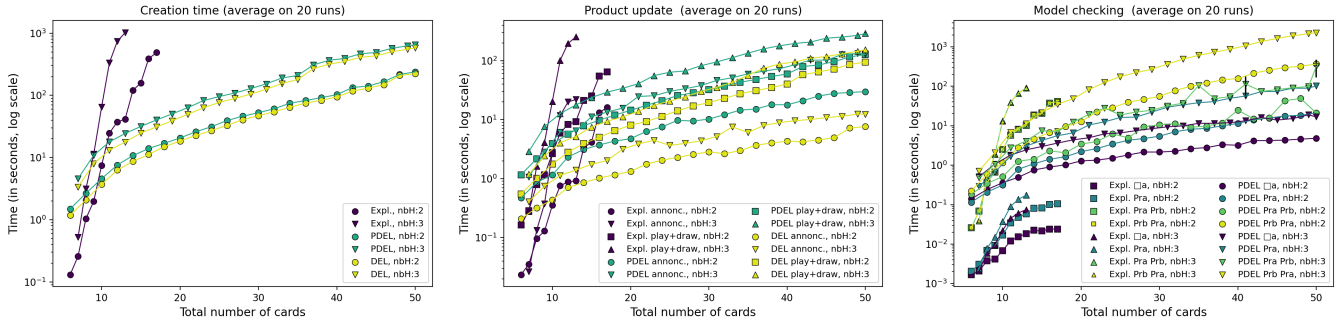
For space reasons, we omit the proof, and stop here without giving the Boolean translation of $[\mathcal{E}_e]$; it is not hard, but rather long, and of secondary interest (update models with full PDEL preconditions have virtually no use in realistic games — as for Hanabi, its update models only use propositional pre- and postconditions).

4 EXPERIMENTS ON HANABI

We now report on experiments we ran on our Python implementation of the PDEL framework, comparing the memory and time efficiency of using "explicit" and symbolic model checking (probabilistic and non-probabilistic), on the game Hanabi (§ 2.1). The concrete language we used to represent PBFs is that of ADDs, which, while strictly less succinct than other decision diagram languages such as SLDDs, has efficient algorithms for almost all the operators we need [12]. We used a (heavily) modified version of the ADD package in pyddlib [6]. Let us now briefly introduce ADDs: *Algebraic decision diagrams* [1] are a generalization to non-Boolean values of the BDD language, in which the two terminal nodes 0 and 1 of BDDs are replaced by as many nodes as necessary. More precisely, an ADD is a directed acyclic graph with one root, where every node N is labelled with a symbol $x \in PS$ and has two outgoing arcs for *then* and *else*, respectively. On any path from the root to a leaf, symbols must be encountered only once and always in the same order; such paths correspond to assignments of Boolean values to all symbols. Figure 2 gives an example of an ADD.

Basic operations on ADDs are polynomial; e.g., applying an operator (or, and, sum, product) between two ADDs is quadratic (it is basically done as an automaton product). Things are not so nice for marginalization or forgetting: they are quadratic in the worst case, for *a single variable*, so they can yield exponential results when one needs to apply them a lot. However, things can still work reasonably well in practice because (1) marginalizations remain polynomial if the variables to be forgotten are at the end of the order, (2) there are a lot of symmetries in the structures, and (3) since we use lotteries and a uniform distribution (which is the most common setting in games), ADDs representing symbolic structures do not need many leaves (only two in our experiments), up to the point when normalization is applied during model checking. The aim of the experiments is precisely to assess whether in practice, at scales sufficient to represent Hanabi, our approach works.

The variable order in ADDs is of crucial importance to get small representations. We use a natural order that arises when compiling formulas, after having experimentally checked that varying the position of variable groups (e.g., putting variables from V closer to V' or to V°) did not seem to yield much better results.



(a) Creation of Kripke structures **(b) Updating knowledge** **(c) Model checking with PEL formulas**

Figure 3: Computation time for creating Kripke structures (left), updating knowledge (center), and model checking, for the explicit and symbolic (with and without probabilities) approaches. "nbH:X" means there are X cards in hand.

Let us now describe our experiments: each one has a a *setting* (explicit PDEL, symbolic PDEL, or symbolic DEL) and two parameters – the total number of cards (nbT, varying from 6 to 50, the latter being the number of cards in the real game) and the number of cards in the hands of each player (nbH, varying from 2 to 3). We always limit the number of players to 2; thus, the number of variables needed to model a Hanabi game only depends on nbT and nbH. Last, we consider a uniform distribution of cards. Each experiment is a sequence of three steps: (1) build the initial Kripke structure and all update models, (2) compute some product updates, (3) perform model checking for a few fixed formulas. In our runs, each of the three steps had a timeout of 2400 seconds. In order to execute different experiments simultaneously, we used a server with 4 AMD Opteron 6282SE 2.6GHz processors, 64 cores and 512G RAM, but no experiment exploited any parallelism or multithreading. Similarly, the RAM capacity was not used in full: as reported by /usr/bin/time, no experiment with nbH = 2 needed more than 8.3G; with nbH = 3, it was 8.7G for 32 cards, 16.7G for 38, 35.7G for 46, and 41.9G for 50. We ran each experiment twenty times, but the variance is very low; standard deviation is indicated as a black vertical line on the graphs when it is more than 5 seconds. All the code needed to reproduce the experiments (and to run unit tests on random structures and formulas) is available online [14].

Fig. 3a shows the creation times of both static Kripke structures and update models. For the explicit case, curves clearly show the combinatorial explosion of the number of worlds. With the symbolic approach, generating the initial Kripke structure of the game with 50 cards is actually quite fast, but generating the update models is not – the reason is that they feature dependencies between many variables present at different positions in the variable order.

Fig. 3b shows the time taken to update the initial knowledge structure by applying actions from the game, implemented as sequences of product updates. We chose two actions: "annonc." is agent $\mathfrak{a}$ announcing to agent $\mathfrak{b}$ that it has a "1" in its hand, and "play+draw" is agent $\mathfrak{a}$ playing its first card and then drawing a new one. The symbolic product update is feasible up to 50 cards, but the explicit version looks like it would already take quite a long time for 30 cards, although it could not be tested on more than 18 cards because of the timeout during the initial creation of structures. We also looked at the evolution of the size of ADDs when applying updates in sequence: roughly, announcements reduce ADD sizes (they remove uncertainty), as do "play" and "discard" actions, whereas "draw" actions increase ADD sizes. Clearly, it is likely that simulating a long sequence of "draw" actions would yield an exponentially large ADD, but in Hanabi at least these actions are virtually always interspersed with uncertainty-reducing actions.

Finally, Fig. 3c shows the time taken to model check each of the following four PEL formulas, where $\phi^1_{\mathfrak{a}}$ is the formula representing "the first card of agent $\mathfrak{a}$ is a 1": $\Box_{\mathfrak{a}}\phi^1_{\mathfrak{a}}$, $\mathrm{Pr}_{\mathfrak{a}}(\phi^1_{\mathfrak{a}}) \geq 0.25$, $\mathrm{Pr}_{\mathfrak{a}}(\mathrm{Pr}_{\mathfrak{b}}(\phi^1_{\mathfrak{a}}) \geq 0.25) \geq 0.25$, and $\mathrm{Pr}_{\mathfrak{b}}(\mathrm{Pr}_{\mathfrak{a}}(\phi^1_{\mathfrak{a}}) \geq 0.25) \geq 0.25$, respectively denoted in the caption by □a, Pra, PraPrb, and PrbPra. We do not show the time taken to model check propositional formulas – it is always virtually instantaneous. Results for explicit structures (when they could have been built) are quite good for formulas of depth 1, not so much for formulas of depth 2; on symbolic structures, the order of the Pr operators has a huge impact, but model checking remains feasible all the way up to 50 cards.

5 CONCLUSION

We presented a symbolic representation of the probabilistic Kripke structures and probabilistic update models of PDEL using pseudo-boolean functions, extending the SMCDEL framework to the management of probabilities. This makes it possible to do model checking and to compute product updates symbolically. Our approach is fully implemented, using ADDs as the data structure representing PBFs. We conducted experiments on the game Hanabi; the results show that our approach scales quite well for near-realistic instances of Hanabi. In future work, we intend to generalize epistemic planning to a probabilistic setting, and to study optimal strategy synthesis for Hanabi and other games with imperfect or incomplete information; our results make us confident that such synthesis could be made feasible in practice.

ACKNOWLEDGMENTS

This work has been partly supported by the Région Normandie, the European Regional Development Fund, and the PING/ACK project of the French National Agency for Research (ANR-18-CE40-0011).

REFERENCES

[1] R. Iris Bahar, Erica A. Frohm, Charles M. Gaona, Gary D. Hachtel, Enrico Macii, Abelardo Pardo, and Fabio Somenzi. 1997. Algebraic Decision Diagrams and Their Applications. *Formal Methods in System Design* 10, 2/3 (1997), 171–206. https://doi.org/10.1023/A:1008699807402

[2] Nolan Bard, Jakob N. Foerster, Sarath Chandar, Neil Burch, Marc Lanctot, H. Francis Song, Emilio Parisotto, Vincent Dumoulin, Subhodeep Moitra, Edward Hughes, Iain Dunning, Shibl Mourad, Hugo Larochelle, Marc G. Bellemare, and Michael Bowling. 2020. The Hanabi challenge: A new frontier for AI research. *Artif. Intell.* 280 (2020), 103216. https://doi.org/10.1016/j.artint.2019.103216

[3] Antoine Bauza. 2010. Hanabi. http://www.antoinebauza.fr/?tag=hanabi. accessed 2018-06-11.

[4] Thomas Bolander. 2017. A Gentle Introduction to Epistemic Planning: The DEL Approach. *Electronic Proceedings in Theoretical Computer Science* 243 (Mar 2017), 1–22. https://doi.org/10.4204/eptcs.243.1

[5] Randal E. Bryant. 1986. Graph-Based Algorithms for Boolean Function Manipulation. *IEEE Trans. Computers* 35, 8 (1986), 677–691. https://doi.org/10.1109/TC.1986.1676819

[6] Thiago Pereira Bueno. 2017. pyddlib, a Python3 library for manipulating decision diagrams. https://github.com/thiagopbueno/pyddlib/

[7] Tristan Charrier, Sophie Pinchinat, and François Schwarzentruber. 2019. Symbolic model checking of public announcement protocols. *J. Log. Comput.* 29, 8 (2019), 1211–1249. https://doi.org/10.1093/logcom/exz023

[8] Tristan Charrier and François Schwarzentruber. 2017. A Succinct Language for Dynamic Epistemic Logic. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems, AAMAS 2017, São Paulo, Brazil, May 8-12, 2017*, Kate Larson, Michael Winikoff, Sanmay Das, and Edmund H. Durfee (Eds.). ACM, 123–131. http://dl.acm.org/citation.cfm?id=3091148

[9] Adnan Darwiche and Pierre Marquis. 2002. A Knowledge Compilation Map. *J. Artif. Intell. Res.* 17 (2002), 229–264. https://doi.org/10.1613/jair.989

[10] Lorenz Demey and Barteld Kooi. 2014. Logic and Probabilistic Update. In *Johan van Benthem on Logic and Information Dynamics*, Alexandru Baltag and Sonja Smets (Eds.). Springer, 381–404. https://doi.org/10.1007/978-3-319-06025-5_13

[11] Ronald Fagin and Joseph Y. Halpern. 1994. Reasoning About Knowledge and Probability. *J. ACM* 41, 2 (1994), 340–367. https://doi.org/10.1145/174652.174658

[12] Hélène Fargier, Pierre Marquis, Alexandre Niveau, and Nicolas Schmidt. 2014. A Knowledge Compilation Map for Ordered Real-Valued Decision Diagrams. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, July 27 -31, 2014, Québec City, Québec, Canada*, Carla E. Brodley and Peter Stone (Eds.). AAAI Press, 1049–1055. http://www.aaai.org/ocs/index.php/AAAI/AAAI14/paper/view/8195

[13] Hélène Fargier, Pierre Marquis, and Nicolas Schmidt. 2013. Semiring Labelled Decision Diagrams, Revisited: Canonicity and Spatial Efficiency Issues. In *IJCAI 2013, Proceedings of the 23rd International Joint Conference on Artificial Intelligence, Beijing, China, August 3-9, 2013*, Francesca Rossi (Ed.). IJCAI/AAAI, 884–890. http://www.aaai.org/ocs/index.php/IJCAI/IJCAI13/paper/view/6623

[14] Sébastien Gamblin, Alexandre Niveau, and Maroua Bouzid. 2022. Reproduction Package for "A Symbolic Representation for Probabilistic Dynamic Epistemic Logic" (AAMAS 2022). https://doi.org/10.5281/zenodo.5966036

[15] Malvin Gattinger. 2018. *New directions in Model Checking Dynamic Epistemic Logic.* Ph.D. Dissertation. Universiteit van Amsterdam.

[16] Doga Kisa, Guy Van den Broeck, Arthur Choi, and Adnan Darwiche. 2014. Probabilistic Sentential Decision Diagrams. In *Principles of Knowledge Representation and Reasoning: Proceedings of the Fourteenth International Conference, KR 2014, Vienna, Austria, July 20-24, 2014*, Chitta Baral, Giuseppe De Giacomo, and Thomas Eiter (Eds.). AAAI Press. http://www.aaai.org/ocs/index.php/KR/KR14/paper/view/8005

[17] Barteld P. Kooi. 2003. Probabilistic Dynamic Epistemic Logic. *Journal of Logic, Language and Information* 12, 4 (2003), 381–408. https://doi.org/10.1023/A:1025050800836

[18] Bastien Maubert, Sophie Pinchinat, and François Schwarzentruber. 2019. Reachability Games in Dynamic Epistemic Logic. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, Sarit Kraus (Ed.). ijcai.org, 499–505. https://doi.org/10.24963/ijcai.2019/71

[19] Kenneth L. McMillan. 1993. *Symbolic model checking.* Kluwer. https://doi.org/10.1007/978-1-4615-3190-6

[20] Daniel Reifsteck, Thorsten Engesser, Robert Mattmüller, and Bernhard Nebel. 2019. Epistemic Multi-agent Planning Using Monte-Carlo Tree Search. In *KI 2019: Advances in Artificial Intelligence*, Christoph Benzmüller and Heiner Stuckenschmidt (Eds.). Springer International Publishing, Cham, 277–289.

[21] Scott Sanner and David A. McAllester. 2005. Affine Algebraic Decision Diagrams (AADDs) and their Application to Structured Probabilistic Inference. In *IJCAI-05, Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence, Edinburgh, Scotland, UK, July 30 - August 5, 2005*, Leslie Pack Kaelbling and Alessandro Saffiotti (Eds.). Professional Book Center, 1384–1390. http://ijcai.org/Proceedings/05/Papers/1439.pdf

[22] Afsaneh Shirazi and Eyal Amir. 2008. Factored Models for Probabilistic Modal Logic. In *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence, AAAI 2008, Chicago, Illinois, USA, July 13-17, 2008*, Dieter Fox and Carla P. Gomes (Eds.). AAAI Press, 541–547. http://www.aaai.org/Library/AAAI/2008/aaai08-086.php

[23] Johan van Benthem, Jelle Gerbrandy, and Barteld P. Kooi. 2009. Dynamic Update with Probabilities. *Studia Logica* 93, 1 (2009), 67–96. https://doi.org/10.1007/s11225-009-9209-y

[24] Johan van Benthem, Jan van Eijck, Malvin Gattinger, and Kaile Su. 2018. Symbolic model checking for Dynamic Epistemic Logic — S5 and beyond. *Journal of Logic and Computation* 28, 2 (11 2018), 367–402. https://doi.org/10.1093/logcom/exx038 arXiv:http://oup.prod.sis.lan/logcom/article-pdf/28/2/367/24261865/exx038.pdf

[25] Johan van Benthem, Jan van Eijck, and Barteld P. Kooi. 2006. Logics of communication and change. *Inf. Comput.* 204, 11 (2006), 1620–1662. https://doi.org/10.1016/j.ic.2006.04.006

[26] Hans van Ditmarsch, Wiebe van der Hoek, and Barteld Kooi. 2007. *Dynamic Epistemic Logic* (1st ed.). Springer Publishing Company, Incorporated.

[27] Jan van Eijck and François Schwarzentruber. 2014. Epistemic Probability Logic Simplified. In *Advances in Modal Logic 10, invited and contributed papers from the tenth conference on "Advances in Modal Logic," held in Groningen, The Netherlands, August 5-8, 2014*, Rajeev Goré, Barteld P. Kooi, and Agi Kurucz (Eds.). College Publications, 158–177. http://www.aiml.net/volumes/volume10/Eijck-Schwarzentruber.pdf

[28] Nic Wilson. 2005. Decision Diagrams for the Computation of Semiring Valuations. In *IJCAI-05, Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence, Edinburgh, Scotland, UK, July 30 - August 5, 2005*, Leslie Pack Kaelbling and Alessandro Saffiotti (Eds.). Professional Book Center, 331–336. http://ijcai.org/Proceedings/05/Papers/0857.pdf