Parameterized Complexity of Dynamic Belief Updates

Thomas Bolander[1] and Arnaud Lequen[2]

[1] DTU Compute, Technical University of Denmark, tobo@dtu.dk
[2] Univ Rennes, ENS Rennes, France, arnaud.lequen@ens-rennes.fr

Abstract. Dynamic Belief Update (DBU) is a model checking problem in Dynamic Epistemic Logic (DEL) concerning the effect of applying a number of epistemic actions on an initial epistemic model. It can also be considered as a plan verification problem in epistemic planning. The problem is known to be PSPACE-hard. To better understand the source of complexity of the problem, previous research has investigated the complexity of 128 parameterized versions of the problem with parameters such as number of agents and size of actions. The complexity of many parameter combinations has been determined, but previous research left a few combinations as open problems. In this paper, we solve most of the remaining open problems by proving all of them to be fixed-parameter intractable. Only two parameter combinations are still left as open problem for future research.

Keywords: Parameterized Complexity · Model Checking · Dynamic Epistemic Logic · Plan Verification

1 Introduction

In the fields of psychology, ecology, economy, and various areas of computer science like automated planning and distributed systems, the need often arises to model multi-agent systems and reason about the knowledge of the involved agents. Indeed, situations where multiple human or artificial agents interact with their environment, and have to update their knowledge accordingly, are ubiquitous. Dynamic Epistemic Logic (DEL) is a well-suited framework to model such situations, as it is a family of modal logics that allow not only to reason about (higher-order) knowledge, but also to represent how such knowledge is dynamically updated through the occurrence of events. Unfortunately, many decision problems associated with DEL are provably hard [10,6]. Despite that, in real-life situations humans manage to reason fairly effectively about the knowledge of themselves and other agents (at least to modest depths of reasoning). Moreover, certain tasks involving DEL can be carried out fairly easily [10].

In this paper, we study the Dynamic Belief Update (DBU) problem, which boils down to verifying whether an epistemic formula holds in a model after a series of epistemic updates, *i.e.*, whether a certain epistemic fact holds after a sequence of (epistemic) events have occurred in an initial (epistemic) situation.

The events can also be thought of as actions executed by agents, and hence DBU can equivalently be thought of as a plan verification problem in an epistemic setting. We extend the efforts of van de Pol *et al.* [10] to identify which aspects of DBU make it intractable. Of the set of sub-problems of DBU identified by van de Pol *et al.*, we manage to settle the tractability question of most problems previously left open, leaving only two undecided.

In section 2, we present the DEL framework of this paper, and after recalling notions of parameterized complexity, we present DBU and its parameters. In section 3, we prove our new fixed-parameter intractability results of DBU.

2 Background

2.1 Dynamic Epistemic Logic

Dynamic Epistemic Logic (DEL) is a modal logic focused on reasoning about knowledge, which can be revised according to the evolution of the situation [6]. In this paper, we use a variant of DEL that allows multi-pointed epistemic models and has propositional postconditions [3]. The language $\mathcal{L}_K(P, \mathcal{A})$ of multi-agent epistemic logic is defined as follows, where p ranges over a finite set of propositional variables P, and i over a finite set of agents $\mathcal{A}$:

$$\varphi := \top \mid p \mid \neg\varphi \mid \varphi \wedge \varphi \mid K_i\varphi,$$

The intended meaning of $K_i\varphi$ is "agent i knows φ". We will often use the abbreviated notation $\hat{K}_i\varphi = \neg K_i \neg\varphi$, which reads "agent i considers φ possible". Other symbols such as $\vee$ and $\rightarrow$ can be defined by abbreviation as usual. The semantic of the language is defined through *epistemic models* (Kripke models).

Definition 1. (Pointed Epistemic Model) *A* pointed epistemic model *for the language $\mathcal{L}_K(P, \mathcal{A})$ is a pair $(\mathcal{M}, W_d)$ where $\mathcal{M} = (W, R, V)$ and:*

- *W is a finite, non-empty set of worlds*
- *$W_d \subseteq W$ is the non-empty set of the* designated worlds
- *$R : \mathcal{A} \rightarrow 2^{W \times W}$ is a function assigning an equivalence relation R_i to every agent i, called the* indistinguishability relation *for agent i*
- *$V : P \rightarrow 2^W$ is a valuation function that assigns to every propositional variable the set of worlds in which it is true*

Definition 2. (Truth in a pointed epistemic model) *Let $(\mathcal{M}, W_d)$ be a pointed epistemic model, where $\mathcal{M} = (W, R, V)$, and let $\varphi \in \mathcal{L}_K(P, \mathcal{A})$, and $w \in W$. The truth conditions for φ are the standard propositional ones plus:*

$$\begin{array}{lll} (\mathcal{M}, \{w\}) \models K_i\varphi & \textit{iff} & \textit{for all } w' \textit{ s.t. } R_i(w, w'),\ (\mathcal{M}, \{w'\}) \models \varphi \\ (\mathcal{M}, W_d) \models \varphi & \textit{iff} & \textit{for all } w \in W_d,\ (\mathcal{M}, \{w\}) \models \varphi \end{array}$$

Example 1. Figure 1 shows an epistemic model where agent i can not make the distinction between worlds w_1 and w_2. Thus, it does not know whether p is true or not, as it holds in the "actual" world w_1, but not in w_2. As such, $(\mathcal{M}, \{w_1\}) \not\models K_i p$, although $(\mathcal{M}, \{w_1\}) \models p$. As q is true in both worlds, $(\mathcal{M}, \{w_1\}) \models K_i q$.

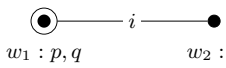

Fig. 1: A pointed epistemic model $(\mathcal{M}, \{w_1\})$ for $\mathcal{L}_K(\{p,q\},\{i\})$ with $\mathcal{M} = (W, R, V)$, $W = \{w_1, w_2\}$, $R_i = \{(w_1,w_1),(w_1,w_2),(w_2,w_1),(w_2,w_2)\}$ and $V(p) = \{w_1\}$, $V(q) = \{w_1, w_2\}$. Reflexive edges are generally omitted.

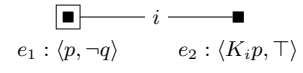

Fig. 2: A pointed event model $(\mathcal{E}, \{e_1\})$ for $\mathcal{L}_K(\{p,q\},\{i\})$ with $\mathcal{E} = (E, Q, \mathsf{pre}, \mathsf{post})$, $E = \{e_1, e_2\}$, $Q_i = \{(e_1,e_1),(e_1,e_2),(e_2,e_1),(e_2,e_2)\}$, $\mathsf{pre}(e_1) = p$, $\mathsf{pre}(e_2) = K_i p$, $\mathsf{post}(e_1) = \neg q$ and $\mathsf{post}(e_2) = \top$.

Event models, defined next, represent changes to the situation, which lead agents to update their knowledge.

Definition 3. (Pointed Event Model) *A* pointed event model *for* $\mathcal{L}_K(P,\mathcal{A})$ *is a pair* $(\mathcal{E}, E_d)$ *where* $\mathcal{E}$ *is a tuple* $\mathcal{E} = (E, Q, \mathsf{pre}, \mathsf{post})$*, such that*

- E *is a non-empty finite set of* events
- $E_d \subseteq E$ *is a non-empty set of* designated events
- $Q : \mathcal{A} \to 2^{E\times E}$ *is a function assigning an equivalence relation* Q_i *to every agent* i*, called the* indistinguishability relation *for agent* i
- $\mathsf{pre} : E \to \mathcal{L}_K(P,\mathcal{A})$ *is a function assigning to each event a precondition*
- $\mathsf{post} : E \to \mathcal{L}_K(P,\mathcal{A})$ *is a function assigning to each event a postcondition, which is a conjunction of literals (propositional variables and their negations, including* $\top$*)*

When no confusion can arise, we will use the abbreviated notation $\mathcal{M}$ for pointed epistemic models $(\mathcal{M}, W_d)$, and similarly for pointed event models. Epistemic models can be updated with the application of event models through *product updates*, defined as follows.

Definition 4. *The product update of the (pointed) epistemic model* $(\mathcal{M}, W_d)$ *with the (pointed) event model* $(\mathcal{E}, E_d)$ *is the (pointed) epistemic model* $(\mathcal{M}, W_d)\otimes (\mathcal{E}, E_d) = (\mathcal{M}', W_d')$*, such that* $\mathcal{M}' = (W', R', V')$ *and*

- $W' = \{(w,e) \in W \times E \mid \mathcal{M}, w \models \mathsf{pre}(e)\}$
- $R_i' = \{((w,e),(v,f)) \in W' \times W' \mid R_i(w,v)$ *and* $Q_i(e,f)\}$
- $V'(p) = (\{(w,e) \in W' \mid \mathcal{M}, w \models p\} \cup \{(w,e) \in W' \mid \mathsf{post}(e) \models p\}) - \{(w,e) \in W' \mid \mathsf{post}(e) \models \neg p\}$
- $W_d' = \{(w,e) \in W' \mid w \in W_d$ *and* $e \in E_d\}$

Example 2. Figure 2 shows an event model where event e_1 or e_2 can occur, and agent i cannot distinguish which event actually happens. Event e_1 can only occur in worlds where p is true, and updates them by making q false. Event e_2 can only occur in worlds where agent i knows p, and does not change the truth value of any variable. If we take the product update $(\mathcal{M}, \{w_1\}) \otimes (\mathcal{E}, \{e_1\})$ of the epistemic model of Figure 1 with the event model of Figure 2, we get a model containing only a single world satisfying $p \wedge \neg q$: the only world satisfying any of the event preconditions is w_1 and it only satisfies the precondition of e_1. So only the world-event pair (w_1, e_1) "survives" the product update, and the postcondition of e_1 enforces q to become false (but otherwise preserves the truth-values from w_1).

2.2 Parameterized Complexity

In this section, we recall some notions of parameterized complexity. Parameterized complexity is a branch of complexity theory whose aim is to offer a finer-grained analysis of a computational problem, taking into account some characteristics of each instance. It studies *parameterized problems*, which resemble classical decision problems. Given an alphabet Σ, a parameterized problem L is a subset of $\Sigma^* \times \mathbb{N}$. Given an instance $\langle x, k\rangle$ of L, we call x the *main part* and k the *parameter*. The parameter k is a metric that gauges one dimension of x. For instance, if our problem is to model-check formulas of $\mathcal{L}_K(P, \mathcal{A})$, then x consists of a formula ϕ and a model $\mathcal{M}$, while k can e.g. be the modal depth of ϕ or the number of agents mentioned in ϕ and $\mathcal{M}$.

In classical complexity theory, the class of tractable problems is P. The corresponding class in parameterized complexity theory is the class of *fixed-parameter tractable* problems, which is denoted FPT. It encompasses all parameterized problems that can be solved by an *fpt-algorithm*, defined as follows.

Definition 5. (Fpt-algorithm) *Let L be a parameterized problem. An algorithm $\mathbb{A}$ is an* fpt-algorithm *for problem L if there exists a computable function $f : \mathbb{N} \to \mathbb{N}$ and a polynomial $\mathcal{P}$, such that the running time of $\mathbb{A}$ on any instance $\langle x, k\rangle \in L$ is at most*

$$f(k) \cdot \mathcal{P}(|x|)$$

For instance, the problem SAT is notoriously intractable [5]. However, its parameterized variant p-SAT, where p is the number of propositional variables, is fixed-parameter tractable. Indeed, checking all 2^{p} assignments of the p variables against a formula φ can be done in time $2^{\mathsf{p}} \cdot \mathcal{P}(|\varphi|)$, for some polynomial $\mathcal{P}$. Intuitively, this means that a set of instances of SAT, where all formulas have a number of variables bounded by some constant p, forms a tractable problem.

Proving that a parameterized problem is not fixed-parameter tractable can be done through *fpt-reductions*, defined next. They can be seen as the parameterized complexity counterpart of classical polynomial reductions, and are useful for proving membership and hardness results for parameterized problems.

Definition 6. (Fpt-reduction) *Let L and L' be two parameterized problems. An* fpt-reduction *from L to L' is a mapping $R : L \longrightarrow L'$ such that:*

- *$\langle x, k\rangle \in L$ iff $\langle x', k'\rangle = R(\langle x, k\rangle) \in L'$.*
- *R is computable by an fpt-algorithm,* i.e.*, there is a computable function f and a polynomial $\mathcal{P}$ such that $R(\langle x, k\rangle)$ can be computed in time $f(k) \cdot \mathcal{P}(|x|)$.*
- *There exists a polynomial g such that, if $\langle x, k\rangle \in L$ and $\langle x', k'\rangle = R(\langle x, k\rangle) \in L'$, then $k' \leq g(k)$.*

When there exists an fpt-reduction from L to L', we write $L \leq_{fpt} L'$.

Suppose $L \leq_{fpt} L'$. It follows from the way fpt-reductions are defined that if L' belongs to some complexity class $\mathcal{C}$, then so does L [8]. Hence, to prove that a problem L' is not fixed-parameter tractable, it suffices to find an fpt-reduction

Dynamic Belief Update (DBU)

Input: An epistemic model $(\mathcal{M}, W_d)$ on $\mathcal{L}_K(P, \mathcal{A})$;
A series of event models $(\mathcal{E}_1, E_1), \ldots, (\mathcal{E}_u, E_u)$ on $\mathcal{L}_K(P, \mathcal{A})$;
A goal formula $\varphi_g \in \mathcal{L}_K(P, \mathcal{A})$.

Output: *Yes* if $(\mathcal{M}, W_d) \otimes (\mathcal{E}_1, E_1) \otimes \cdots \otimes (\mathcal{E}_u, E_u) \models \varphi_g$
No otherwise

Fig. 3: The decision problem DBU considered in this paper

to L' from a problem L known to be not fixed-parameter tractable (we call such problems *fixed-parameter intractable*). In this paper, we consider two complexity classes that are deemed fixed-parameter intractable, namely W[1] and para-NP [7]. W[1] is defined as the class of problems that can be fpt-reduced to k-W2SAT, which is the problem where, given a 2CNF formula φ and a parameter k, one has to decide if there exists a valuation satisfying φ in which at most k variables are true. Para-NP is the class of parameterized problems that can be solved in polynomial time by a nondeterministic fpt-algorithm. Para-NP-hard problems are deemed fixed-parameter intractable, as W[1] $\subseteq$ para-NP [8].

In the remaining of this paper, we will allow problems to have multiple parameters. If a problem L has a set of parameters $\{\mathsf{k}_1, \ldots, \mathsf{k}_n\}$, then its instance are of the form $\langle x, \mathsf{k}_1 + \cdots + \mathsf{k}_n \rangle$. A problem L with parameters $\{\mathsf{k}_1, \ldots, \mathsf{k}_n\}$ is often denoted $\{\mathsf{k}_1, \ldots, \mathsf{k}_n\}$-$L$. When adding further parameters to a parametrized problem, we of course make it more constrained. That is, for any problem L and parameter sets X and Y, the problem $(X \cup Y)$-L is at least as constrained as X-L. Hence the following is easily proved.

Proposition 1. *Let X and Y be sets of parameters of a decision problem L. Then $(X \cup Y)$-$L \leq_{fpt} X$-L.*

2.3 Dynamic Belief Update

The decision problem considered in this paper is presented in Figure 3, following van de Pol *et al.* [10]. It is the problem of checking whether a certain epistemic formula is true after having updated an initial epistemic situation (epistemic model) with a sequence of epistemic actions (event models).[3] So it is about the complexity of keeping track of "who knows what" when observing a sequence of actions taking place, where these actions can both change ontic facts and what the different agents know. Such problems occur e.g. in the coordinated attack problem, the consecutive number puzzle, the muddy children puzzle, board

[3] A better name would probably be "Dynamic Knowledge Update" as we are here only considering models where the underlying accessibility relations are equivalence relations (*i.e.*, S5). However, since all our results are intractability results, these still hold if we generalise to arbitrary accessibility relations, including ones representing beliefs.

Param.	Description	Param.	Description
a	Number of agents	o	Goal formula's modal depth
c	Max. length of event preconditions	p	Number of prop. variables
e	Max. no. of events per event model	u	Number of event models
f	Length of goal formula		

Table 1: Parameters for DBU

Param. for DBU	Complexity
{a, c, f, o, u}	W[1]-hard
{a, f, o, p, u}	W[1]-hard
{e, u}	FPT

Earlier known results [10]

Param. for DBU	Complexity
{a, c, e, f, o, p}	para-NP-hard
{c, f, o, p, u}	W[1]-hard
{a, c, o, p, u}	W[1]-hard

New results of this paper

Table 2: Complexity results for the most general parameterized variants of DBU, from which all other results for our set of parameters can be immediately deduced. Results on the left table originate from [10], while results on the right table constitute the original contributions of this paper.

games like Hanabi and Clue and the false-belief tasks studied in cognitive psychology [4,2,1]. We can also think of the problem as the plan verification problem in epistemic planning [3]: Given an initial state (epistemic model), a sequence of actions (event models) and a goal formula, does the action sequence achieve the goal from the initial state?

DBU is PSPACE-complete, as proven by van de Pol *et al.* [10]. Their paper proposes various parameters as an attempt to identify the mechanisms that make DBU hard. Those parameters are given in Table 1, and any combination of those form a parameterized version of DBU. This leads us to the class of problems of the form X-DBU, where X is a subset of the 7 parameters. For instance, {a, c, p}-DBU is the dynamic belief update problem where the parameters are the number of agents, the length of the preconditions and the number of propositional variables. There are $2^7 = 128$ problems of this form. Prior to our work, the (fixed-parameter) tractability or intractability of 114 of them was already known [10]. We show intractability results for an additional 12 problems, thus leaving only 2 (closely related) problems unsettled. Table 2 summarizes the known results, including the new ones of this paper. It only mentions the strongest ones, as all other results can be immediately deduced from them through Proposition 1, and the observation that, for any set of parameters X of DBU, $(X \cup \{\mathsf{f}\})$-DBU $\leq_{fpt}$ $(X \cup \{\mathsf{f}, \mathsf{o}\})$-DBU (if we constrain the length of the goal formula, we are also constraining its modal depth).

It can be hard to keep track of 128 different versions of the same problem. However, many are obviously interdependent in the sense that the (in)tractability of one immediately implies the (in)tractability of the other, e.g. through Proposition 1. To keep track of dependency and which problems are still open, we developed a small script, which can be found at `https://github.com/arnaudlequen/dbuproblemfinder`. The script allowed us to find the open problems that would

solve most other open problems, and keeping track of the remaining open problems as we gradually settled more cases.

3 Complexity results

Theorem 1. *$\{a, c, e, f, o, p\}$-DBU is fixed-parameter intractable (more precisely, para-NP-hard). In other words, the Dynamic Belief Update problem is intractable even when restricting the number of propositional variables and agents (p,a), the maximum number of events in event models (e), the maximum length of event preconditions (c), and the length and modal depth of the goal formula (f,o).*

Proof. The proof is by an fpt-reduction from an NP-hard problem to an instance of $\{\mathsf{a}, \mathsf{c}, \mathsf{e}, \mathsf{f}, \mathsf{o}, \mathsf{p}\}$-DBU with fixed values of a, c, e, f, o and p, thus proving para-NP-hardness of the latter (since the NP-hard problem doesn't have any parameter, the reduction is also a regular polynomial reduction). The construction used in the proof is an adaptation of the proof of Theorem 19 of Bolander *et al.* [3]. The general idea is to simulate, through an instance of DBU, the execution of a fixed nondeterministic Turing machine M that solves a given NP-hard problem (any NP-hard problem will do). We begin by encoding the initial configuration of the machine (*i.e.*, its tape, the position of its head and its internal state) into the initial epistemic model. Then, we build a series of event model updates, such that the epistemic model after n product updates contains the representation of every configuration of M that can be reached in exactly n transitions (computation steps). Finally, we build a goal formula that checks whether an accepting configuration was encountered in the process or not. Thus, the DBU instance is positive if and only if M accepts the word in the input.

Let $M = (\mathcal{S}, \Gamma, q_0, \delta, q_f)$ be any nondeterministic Turing machine that solves an NP-hard problem in polynomial time, with states $\mathcal{S} = \{q_0, q_1, \ldots, q_f\}$, where q_0 is the only initial state, q_f is the only accepting state, Γ is the set of tape symbols including the blank symbol # and δ is the transition function [9]. The DBU instance we build has agents $\mathcal{A} = \{i, j, k, g\}$ and propositional variables $P = \Gamma \cup \mathcal{S} \cup \{r_i, r_j, t\}$. Information cells for agent k (*i.e.*, sets $W_k \subseteq W$ of maximum size that are closed under R_k) are used to encode configurations of M, and agents i and j are used to distinguish the right and the left of each cell of the tape that we encode. We will in all epistemic models enforce $R_k = R_i \cup R_j$ by having $R_k = R_i \cup R_j$ in the initial model, and $Q_k = Q_i \cup Q_j$ in all event models. We will similarly enforce R_g to be the universal relation—*i.e.*, make any two worlds indistinguishable—by making all pairs of worlds in the initial model indistinguishable, and by making all pairs of events of all event models indistinguishable. For simplicity, the R_k and R_g indistinguishability relations will not be explicitly drawn. Furthermore, the reflexive and transitive closure of all indistinguishability relations drawn is implicitly assumed.

A configuration of the machine can be represented by an Instantaneous Description (ID) [9]. Following Bolander *et al.* [3], we represent IDs by epistemic models as illustrated in Figure 4. This pair of information cells for agent k offers

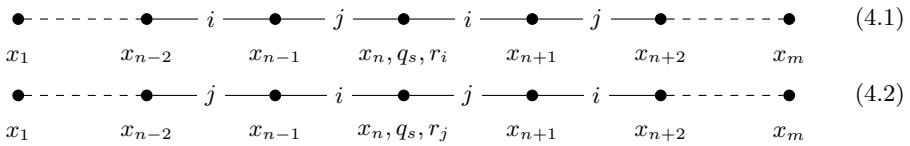

Fig. 4: Two information cells for agent k, both representing the ID $x_1 \cdots x_{n-1} q_s x_n \cdots x_m$ of the Turing machine $M = (\mathcal{S}, \Gamma, q_0, \delta, q_f)$, where $x_i \in \Gamma$ and $q_s \in \mathcal{S}$. This ID represents the configuration of M where the word on the tape is $x_1 \cdots x_m$, where M is in state q_s, and the head is at the nth symbol x_n of the word on the tape. Recall that $R_k = R_i \cup R_j$ and $R_g = W \times W$ is implicitly assumed, where W is the set of all worlds.

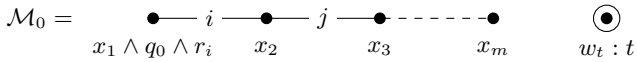

Fig. 5: The initial epistemic model $\mathcal{M}_0$ for the Turing machine M with input word $\omega = x_1 \cdots x_m$. It consists of the represented ID of the initial configuration of M plus an additional designated world w_t only accessible from the other worlds by the R_g relation (recall that $R_k = R_i \cup R_j$ and $R_g = W \times W$ is implicitly assumed).

two unique representations of an ID [3], and we call *represented ID* an information cell for k that has the form of either (4.1) or (4.2). Each world represents one cell of the tape of the machine, and is marked with a propositional variable representing the symbol in the cell. One world is marked with two additional propositions: one for the current state of the machine (q_s), as well as either r_i or r_j. This world represents the current position of the head and is called the *current world*. The propositions r_i and r_j are used to distinguish between the right and the left of the current cell. If r_i (resp. r_j) is true, then the cell at the right of the current one is reachable through an i-edge (resp. j-edge).

We proceed to show how to build the initial epistemic model and event models. Suppose that in its initial configuration, M is in state q_0 and with the word $\omega = x_1 \cdots x_m$ on its tape. Then the initial epistemic model $\mathcal{M}_0$ is the represented ID of the initial configuration of M, as shown in Figure 5. In addition to that, we add a designated world w_t only labeled by the prop. variable t. Its purpose is to make sure the model doesn't end up being empty, which could otherwise happen if at some point no transition can be applied to any ID.

The next step consists in building the series of event models, which are all copies of a single model $\mathcal{E}_{trans}$. The aim of $\mathcal{E}_{trans}$ is to simulate one step of M, by applying all applicable transitions to each represented ID of the previous epistemic model. The event model mainly consists in a disjoint union of several sub-event models, that we call *transition components*, whose purpose is to attempt to apply a transition of the Turing machine M to a represented ID. For each transition l, *i.e.*, each element of the transition function δ, we construct an i-transition component τ_l^i and a j-transition component τ_l^j. We construct these transition components such that given an ID s and valid transition l for s, applying τ_l^i (resp. τ_l^j) to the represented ID of s, of the form (4.1) (resp. (4.2)), will result in the represented ID of the successor of s after l was applied. Applying

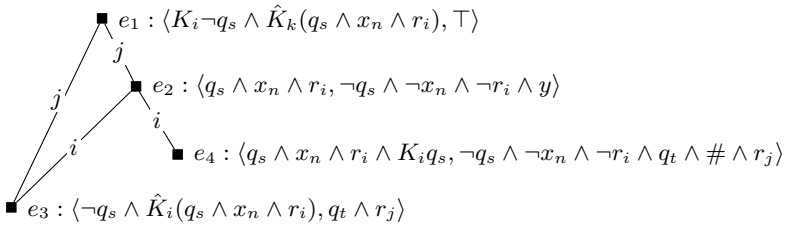

Fig. 6: The transition component τ_l^i, for a transition l of the form $\delta(q_s, x_n) = (q_t, y, R)$, where $x_n \neq y$.

Fig. 7: Event model σ. The purpose of e_f is to carry to the updated model any world marked with q_f, as it means that an accepting configuration has been reached. Event e_t copies the world w_t, as the only designated event.

to an ID s a transition component whose form does not match the represented ID of s, or whose transition is not applicable to s, will yield no worlds.

Figure 6 shows an example of an i-transition component. The j-transition component can be obtained by swapping i and j everywhere. Other transitions, such as $\delta(q_s, x_n) = (q_t, y, L)$ or transitions satisfying $x_n = y$, can be handled similarly. Let us try to explain the intuition behind this construction. It is very similar to the construction of Bolander *et al.* [3]. Event e_1 makes sure that, after the update, worlds that represent cells of the tape that are unaffected by the transition are left unchanged. It copies into the updated model every world of the represented ID, except the world representing the current head position and the one at its right. Event e_2 copies the current world, noted w, but removes the propositional variables that mark the head of the machine. It also updates the tape symbol. If the cell on the right of the current position of the head is not blank, then there exists a world w' on the right of the current world w, *i.e.*, such that $R_i(w, w')$. Event e_3 adds on w' the propositional variables that make it the current world of the updated model. It updates as well the current state of the machine, from q_s to q_t. If the cell on the right of the current position of the head is blank, then no world is on the right of the current world. Event e_4 creates it with a blank symbol, and sets it to be the current world of the updated model. Applying the i-transition component of Figure 6 to a represented ID s of the form (4.2) results in no world. Indeed, in s, the current world is instead labeled by r_j, and thus, no world verifies r_i. Therefore, no event has its precondition satisfied, as each of the four events $e_1, \ldots, e_4$ has a precondition requiring r_i to hold in at least one world. Similarly, if the transition is not applicable to the ID represented by s, then the current world of s is labeled by $q'_s \neq q_s$ and/or $x'_n \neq x_n$, and thus does not satisfy $q_s \wedge x_n$. And as before, each of the four events $e_1, \ldots, e_4$ has a precondition requiring $q_s \wedge x_n$ to hold in at least one world.

In order to build $\mathcal{E}_{trans}$, we need to introduce another component σ, which consists of two events, e_f and e_t. Those events, as depicted in Figure 7, carry

to the updated model the information that will eventually allow the goal formula to check whether the instance is positive or not. Building $\mathcal{E}_{trans}$ is then straightforward. In addition to σ, it consists in the disjoint union of the i- and j-transition components τ_l^i and τ_l^j associated to every transition l of M. Recall again that we implicitly assume to also add a g-edge between any pair of events. Applying $\mathcal{E}_{trans}$ to an epistemic model that contains the representations of all IDs reachable in n transitions results in a model containing the representations of all IDs reachable in $n+1$ transitions. If the model contained any world where q_f was true, then in the updated model, there is also a world where q_f if true.

By assumption, there exists a polynomial $\mathcal{P}$ such that, for any word ω', M accepts ω' iff M accepts it in at most $\mathcal{P}(|\omega'|)$ steps. Then, for our given input ω, we only need to simulate $\mathcal{P}(|\omega|)$ steps of M, and thus create a series of $\mathcal{P}(|\omega|)$ product updates of $\mathcal{M}_0$ with the event model $\mathcal{E}_{trans}$. In the final model, the only designated world is w_t, which is linked by a g-edge to every other remaining world. The goal formula $\hat{K}_g q_f$ must thus be true in the final model iff a world verifying q_f has been reach after some initial sequence of product updates, *i.e.*, if M can reach an accepting state in at most $\mathcal{P}(|\omega|)$ steps. Thus, M accepts input ω iff the instance of DBU with initial state $\mathcal{M}_0$, with $\mathcal{P}(|\omega|)$ copies of the event model $\mathcal{E}_{trans}$ and with goal formula $\hat{K}_g q_f$ is positive. We have now fpt-reduced the problem "Does M accept input ω?", where M is fixed and ω is the input, to the problem $\{\mathsf{a},\mathsf{c},\mathsf{e},\mathsf{f},\mathsf{o},\mathsf{p}\}$-DBU. We comply with the conditions of Definition 6: we respectively satisfy the second and third conditions as the reduction is polynomial, and all parameters of $\{\mathsf{a},\mathsf{c},\mathsf{e},\mathsf{f},\mathsf{o},\mathsf{p}\}$-DBU are constants, by construction. In particular, p and e are constants as they only depend on M, which is fixed and not part of the input. Finally, as M solves an NP-hard problem, $\{\mathsf{a},\mathsf{c},\mathsf{e},\mathsf{f},\mathsf{o},\mathsf{p}\}$-DBU is para-NP-hard.

Corollary 1. *$\{a,c,p\}$-DBU, $\{a,c,p,e\}$-DBU and $\{a,c,p,f\}$-DBU are all fixed-parameter intractable.*

The corollary is by Proposition 1. In addition to settling those four open problems, Theorem 1 shows a stronger result, which is that all parameterized versions of DBU that do not have u as a parameter are fixed-parameter intractable. This settles in itself the fixed-parameter intractability of 64 problems, out of the 128 total. It also constitutes an alternative proof of the intractability of three different problems shown separately by van de Pol *et al.* [10], which are $\{\mathsf{a},\mathsf{c},\mathsf{e},\mathsf{f},\mathsf{o}\}$-DBU, $\{\mathsf{c},\mathsf{e},\mathsf{f},\mathsf{o},\mathsf{p}\}$-DBU and $\{\mathsf{a},\mathsf{e},\mathsf{f},\mathsf{o},\mathsf{p}\}$-DBU.

We now prove fixed-parameter intractability of two further problems that were left open by van de Pol *et al.* [10]: $\{\mathsf{c},\mathsf{f},\mathsf{o},\mathsf{p},\mathsf{u}\}$-DBU and $\{\mathsf{a},\mathsf{c},\mathsf{p},\mathsf{u}\}$-DBU. We here show that both are fixed-parameter intractable, which implies the fixed-parameter intractability of $\{\mathsf{c},\mathsf{f},\mathsf{p},\mathsf{u}\}$-DBU and $\{\mathsf{a},\mathsf{c},\mathsf{p}\}$-DBU. Our proofs of both theorems are adaptations of the fixed-parameter intractability proof of $\{\mathsf{c},\mathsf{o},\mathsf{p},\mathsf{u}\}$-DBU by van de Pol *et al.* [10]. In addition to strengthening their construction to be able to generalize their intractability results, we also simplify their construction in a few places. The general point is to show W[1]-hardness by a reduction from the earlier mentioned W[1]-complete problem k-W2SAT:

Given a 2CNF input formula φ and a parameter k, decide whether there exists a valuation satisfying φ in which at most k variables are true.

In the following we assume the variables of φ are named $x_1, \ldots, x_m$. The general trick in constructing an fpt-reduction from k-W2SAT to a parameterized DBU problem is as follows. First we define epistemic (sub)models that can be used to encode propositional valuations over $\{x_1, \ldots, x_m\}$. We call these *valuation gadgets* and use $\mathcal{M}_v$ to denote the valuation gadget encoding the valuation v. The initial model of the DBU instance is then the model $\mathcal{M}_0$ where 0 denotes the valuation with $0(x_i) = 0$ for all i (the valuation that sets every variable false). We then construct an event model that can take any set of valuation gadgets and for each gadget $\mathcal{M}_v$ it constructs m new gadgets $\mathcal{M}_{v[x_1 \mapsto 1]}, \ldots, \mathcal{M}_{v[x_m \mapsto 1]}$ (where $v[x \mapsto t]$ is the mapping that is as v except $v(x) = t$). After updating k times with this event model, we are guaranteed to have gadgets representing all valuations where at most k variables are true. If we have no bound on f, we can now directly use the goal formula of the DBU instance to check that there exists a gadget making φ true. This is what we do for the intractability proof of $\{\mathsf{a}, \mathsf{c}, \mathsf{p}, \mathsf{u}\}$-DBU. If we have a bound on f, as in the intractability proof of $\{\mathsf{c}, \mathsf{f}, \mathsf{o}, \mathsf{p}, \mathsf{u}\}$-DBU, we need to perform product updates with additional event models that mark the gadgets making φ true.

Theorem 2. *$\{\mathsf{c}, \mathsf{f}, \mathsf{o}, \mathsf{p}, \mathsf{u}\}$-DBU is fixed-parameter intractable (W[1]-hard). In other words, the Dynamic Belief Update problem is intractable even when restricting the number of propositional variables (p), the number of event models (u), the maximum length of event preconditions (c), and the length and modal depth of the goal formula (f,o).*

Proof. The main contribution of this proof over the proof of the fixed-parameter intractability of $\{\mathsf{c}, \mathsf{o}, \mathsf{p}, \mathsf{u}\}$-DBU by van de Pol *et al.* [10] is the construction of an additional event model ($\mathcal{E}_\varphi$) that allow us to only consider a goal formula of fixed length (while still preserving the fixed bound on the event preconditions). Let φ and k be given (an instance of k-W2SAT), where φ has variables $var(\phi) = \{x_1, \ldots, x_m\}$. We will now create an instance of DBU that can decide the k-W2SAT instance, *i.e.*, whether there exists a valuation satisfying ϕ and setting at most k variables true. The DBU instance will be using agents $\mathcal{A} = \{1, \ldots, m, a, b\}$. For each valuation v over $var(\varphi)$, we define the gadget $\mathcal{M}_v$ as the star-shaped model with a single root world satisfying proposition r, and for each x_i with $v(x_i) = 0$ it has an outgoing i-edge to a unique world satisfying no propositions. The construction is illustrated for $m = 4$ in Figure 8. Now consider the event model $\mathcal{E}$ illustrated for $m = 4$ in Figure 9. The events with no label are implicitly labelled $\langle \top, \top \rangle$, *i.e.*, they are events that preserve any world to which they are applied. The events labelled $\langle r, \top \rangle$ only apply to the roots of gadgets. When $\mathcal{E}$ is applied to a gadget $\mathcal{M}_v$, it creates m copies of the gadget, where in the first gadget x_1 is made true (by removing the outgoing 1-edge), in the second x_2 is made true (by removing the outgoing 2-edge), etc. These gadgets are furthermore connected by a-edges via their root worlds. When this event model is applied k times to the initial gadget model $\mathcal{M}_0$, we achieve a

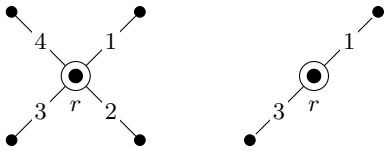

Fig. 8: Left: A valuation gadget for $m = 4$ representing the valuation 0 in which all x_i, $i = 1, \ldots, m$, are false. Right: The gadget for the valuation where x_2 and x_4 are true (since the outgoing 2- and 4-edges have been deleted).

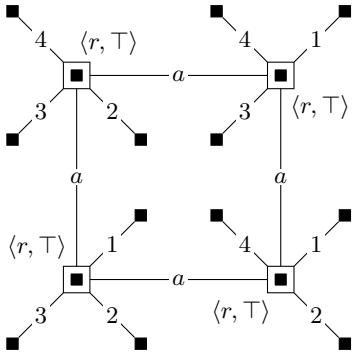

Fig. 9: The pointed event model $\mathcal{E}$ for $m = 4$. The unlabelled events are implicitly labelled $\langle \top, \top \rangle$.

model with m^{k} gadgets connected by a-edges via their root worlds. Each gadget is obtained by starting with the initial gadget representing the valuation 0, and then making at most k variables true by consecutively removing k edges from the gadget model. Since we might attempt to remove the same edge multiple times, this construction gives us a representation of all valuations where at most k variables are true (except the valuation 0 that can be checked separately). Hence the final model $\mathcal{M}_0 \otimes \mathcal{E}^{\mathsf{k}}$ contains a gadget for each valuation with at most k variables set true (except the valuation 0).

Note that a clause $(\neg)x_i \vee (\neg)x_j$ is true in a valuation v iff the formula $(\neg)K_i r \vee (\neg)K_j r$ is true at the root of the gadget $\mathcal{M}_v$. We now construct an additional event model $\mathcal{E}_\varphi$ as follows. It has a single designated event labelled $\langle r, \top \rangle$. For each clause $(\neg)x_i \vee (\neg)x_j$ of φ, it has an additional event labelled $\langle r \wedge \neg((\neg)K_i r \vee (\neg)K_j r), f \rangle$, where f is a new propositional variable denoting "failure". All events of $\mathcal{E}_\varphi$ are connected by b-edges. Each event with postcondition f checks whether a particular clause of φ is false in the gadget to which it is applied. If it is, a b-accessible world satisfying f is created. When $\mathcal{E}_\varphi$ is applied to a valuation gadget, it will hence preserve the root (due to the event $\langle r, \top \rangle$), and additionally it will add a b-accessible f-world for each unsatisfied clause. If there are no unsatisfied clauses, it will only preserve the root. Hence, if we apply $\mathcal{E}_\varphi$ to the model $\mathcal{M}_0 \otimes \mathcal{E}^{\mathsf{k}}$ containing gadgets for all the relevant valuations, the resulting model $\mathcal{M}_0 \otimes \mathcal{E}^{\mathsf{k}} \otimes \mathcal{E}_\varphi$ will contain an r-world with no b-accessible f-worlds iff φ is true in one of the valuations. Hence, we can check whether φ is true in one of the relevant valuations by checking the goal formula $\varphi_g := \hat{K}_a(r \wedge K_b \neg f)$ in the model $\mathcal{M}_0 \otimes \mathcal{E}^{\mathsf{k}} \otimes \mathcal{E}_\varphi$.

To sum up, given a k-W2SAT instance φ with parameter k, we reduce it to the DBU instance with initial model $\mathcal{M}_0$, with k copies of the event model $\mathcal{E}$ followed by the event model $\mathcal{E}_\varphi$ and with goal formula φ_g. We now only have to verify that the reduction is an fpt-reduction from k-W2SAT to $\{\mathsf{c}, \mathsf{f}, \mathsf{o}, \mathsf{p}, \mathsf{u}\}$-DBU. Building the epistemic model $\mathcal{M}_0$ and the k copies of the event model $\mathcal{E}$ is clearly

polynomial in m and k and hence in the input size of the k-W2SAT instance. Building $\mathcal{E}_\varphi$ is polynomial in the formula φ and hence also in the input size of the k-W2SAT instance. Finally, the goal formula has a fixed length. This shows that the reduction is computable by an fpt-algorithm. We then only need to show that the parameters of the translated $\{\mathsf{c},\mathsf{f},\mathsf{o},\mathsf{p},\mathsf{u}\}$-DBU instance can be bound by a computable function in k. The parameters $\mathsf{c},\mathsf{f},\mathsf{o},\mathsf{p}$ all have a fixed value independent of the k-W2SAT instance, and u is $\mathsf{k}+1$. So the parameters are clearly bound by a computable function in k, and the proof is complete.

Theorem 3. *$\{\mathsf{a},\mathsf{c},\mathsf{o},\mathsf{p},\mathsf{u}\}$-DBU is fixed-parameter intractable ($W[1]$-hard).*

Proof. The main contribution of this proof over the proof of the fixed-parameter intractability of $\{\mathsf{c},\mathsf{o},\mathsf{p},\mathsf{u}\}$-DBU by van de Pol *et al.* [10] is that we show how to create gadgets that encode the truth value of the different variables via worlds at different depths of the model rather than via different agents. This is necessary since we have a as a parameter, so we need to put a bound on the number of agents. When refererring to worlds at different depths of a model, and with no bound on the depth of a model, we usually also need preconditions of unbounded length. But our construction shows that it is possible to still do with only preconditions of bounded length.

Essentially, the structure of this proof is as the previous, except we need a different type of gadgets. Let φ and k be given with $var(\varphi) = \{x_1,\dots,x_m\}$. Let $\mathcal{A} = \{1,2,a\}$. For each valuation v, we define the gadget $\mathcal{M}_v$ as an alternating $1,2$-chain of worlds with a root world satisfying r, and where the world at distance i from the root makes t true iff $v(x_i) = 1$. The construction is illustrated for $m = 4$ in Figure 10. Now consider the event model $\mathcal{E}$ illustrated for $m = 4$ in Figure 11. As in the previous proof, when this event model is applied to a gadget $\mathcal{M}_v$, it creates m copies of the gadget, where in the first gadget x_1 is made true (by adding t to the world at distance 1 from the root), in the second x_2 is made true (by adding t to the world at distance 2 from the root), etc. As before, these gadgets will be connected by a-edges via their root worlds. Also as before, when this event model is applied k times to the initial gadget model $\mathcal{M}_0$, we achieve a model with m^{k} gadgets containing at least one gadget for each valuation making at most k variables true (again except the valuation 0 that can be treated separately). The only essential difference is that instead of making use of agents to encode the truth value of the different variables, we use the depth of the event model. This means we can use a as a parameter in our reduction (the number of agents is fixed independently of the input).

Let $\psi_1 := \hat{K}_1 t$, $\psi_2 := \hat{K}_1\hat{K}_2 t$, $\psi_3 := \hat{K}_1\hat{K}_2\hat{K}_1 t$, etc. Then note that φ is true in the valuation v iff the formula $\varphi[\psi_i/x_i]$ is true in the root of the gadget $\mathcal{M}_v$. Hence, to check whether φ is true in a valuation making at most k variables true, we can check whether the formula $\varphi_g := \hat{K}_a\varphi[\psi_i/x_i]$ is true in $\mathcal{M}_0 \otimes \mathcal{E}^{\mathsf{k}}$. To sum up, given a k-W2SAT instance φ with parameter k, we reduce it to the DBU instance with initial model $\mathcal{M}_0$, with k copies of the event model $\mathcal{E}$ and with goal formula φ_g. Building $\mathcal{M}_0$ and the k copies of $\mathcal{E}$ is polynomial in m and k, and building φ_g is polynomial in m and the length of φ. Hence the

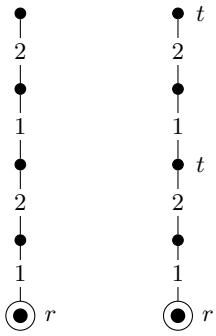

Fig. 10: Left: A valuation gadget for $m = 4$ representing the valuation 0 in which all x_i, $i = 1, \ldots, m$, are false. Right: The gadget for the valuation where x_2 and x_4 are true (since the worlds in distance 2 and 4 from the root have label t).

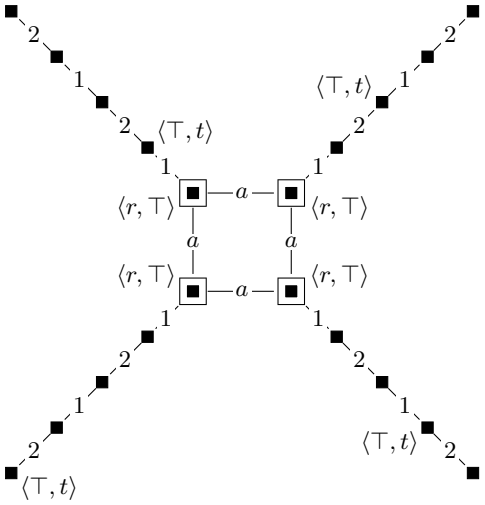

Fig. 11: The pointed event model $\mathcal{E}$ for $m = 4$. The unlabelled events are implicitly labelled $\langle \top, \top \rangle$.

DBU instance can be computed in polynomial time in the size of the k-W2SAT instance, and is hence computable by an fpt-algorithm. We then only need to show that the parameters of the translated $\{\mathsf{a}, \mathsf{c}, \mathsf{o}, \mathsf{p}, \mathsf{u}\}$-DBU instance can be bound by a computable function in k. This trivially holds, as the parameters a, c, o, p all have fixed value independent of the k-W2SAT instance, and u is k.

4 Discussion and future work

We managed to solve most of the open tractability problems for the dynamic belief update problem. In all cases, our results were negative, *i.e.*, we proved fixed-parameter intractability. When entering the new results into our previously mentioned tool, we get that tractability of the following parameter combinations is still open: $\{\mathsf{a}, \mathsf{c}, \mathsf{f}, \mathsf{p}, \mathsf{u}\}$ and $\{\mathsf{a}, \mathsf{c}, \mathsf{f}, \mathsf{o}, \mathsf{p}, \mathsf{u}\}$. We conjecture that both parameter combinations are fixed-parameter tractable, but leave it for future work.

The *short Turing machine acceptance problem* (STMA) is the acceptance problem of nondeterministic Turing machines with bound k on the number of computation steps. It is a parameterized problem with parameter k known to be W[1]-complete, *i.e.*, fixed-parameter intractable [7]. The proof of Theorem 1 gives us a construction allowing us to encode an instance of STMA as a DBU instance. Since the parameter k is the number of computation steps, which translates into the parameter u in the DBU instance, we can do an fpt-reduction from STMA to $\{\mathsf{a}, \mathsf{c}, \mathsf{f}, \mathsf{o}, \mathsf{u}\}$-DBU, *i.e.*, we can replace e, p by u in the fixed-parameter intractability result of Theorem 1. We have to drop the parameters e and p as

their sizes depend on the alphabet of the Turing machine. This reduction then immediately gives W[1]-hardness of $\{\mathsf{a}, \mathsf{c}, \mathsf{f}, \mathsf{o}, \mathsf{u}\}$-DBU. This result was already established by van de Pol *et al.* [10], but with our Turing machine construction in Theorem 1, we get this additional result essentially for free.

In the proof of Theorem 2, we introduced the trick of checking each clause of the 2CNF formula with a single event model, hence allowing us to put a bound on the length of the goal formula. One might be tempted to try out the same trick in the proof of Theorem 3, however that would blow up the length and modal depth of the preconditions, since we need a formula of modal depth i to check whether x_i is true in a valuation gadget. If we found a way to preserve the bound on c, we would achieve a proof of the fixed-parameter intractability of $\{\mathsf{a}, \mathsf{c}, \mathsf{f}, \mathsf{o}, \mathsf{p}, \mathsf{u}\}$-DBU. However, as mentioned, we believe this problem to be tractable.

As future work, we hope to extend our results to epistemic planning, i.e, the problem of plan *synthesis* rather than plan *verification* as considered here, and we would at the same time consider additional relevant parameters.

References

1. Baral, C., Bolander, T., van Ditmarsch, H., McIlrath, S.: Epistemic planning (dagstuhl seminar 17231). In: Dagstuhl Reports. vol. 7. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik (2017)
2. Bolander, T.: Seeing is believing: Formalising false-belief tasks in dynamic epistemic logic. In: Jaakko Hintikka on Knowledge and Game-Theoretical Semantics, pp. 207–236. Springer (2018)
3. Bolander, T., Andersen, M.: Epistemic planning for single- and multi-agent systems. Journal of Applied Non-classical Logics - JANCL **21**, 9–34 (01 2011). https://doi.org/10.3166/jancl.21.9-34
4. Bolander, T., Charrier, T., Pinchinat, S., Schwarzentruber, F.: DEL-based epistemic planning: Decidability and complexity. Artificial Intelligence (2020, to appear). https://doi.org/https://doi.org/10.1016/j.artint.2020.103304, `http://www.sciencedirect.com/science/article/pii/S0004370219301146`
5. Cook, S.A.: The complexity of theorem-proving procedures. In: Proceedings of the Third Annual ACM Symposium on Theory of Computing. p. 151158. STOC 71, Association for Computing Machinery, New York, NY, USA (1971). https://doi.org/10.1145/800157.805047, `https://doi.org/10.1145/800157.805047`
6. Ditmarsch, H.v., van der Hoek, W., Kooi, B.: Dynamic Epistemic Logic. Springer Publishing Company, Incorporated, 1st edn. (2007)
7. Downey, R.G., Fellows, M.R.: Fundamentals of Parameterized Complexity. Springer Publishing Company, Incorporated (2013)
8. Flum, J., Grohe, M.: Parameterized Complexity Theory (Texts in Theoretical Computer Science. An EATCS Series). Springer-Verlag, Berlin, Heidelberg (2006)
9. Hopcroft, J.E., Motwani, R., Ullman, J.D.: Introduction to Automata Theory, Languages, and Computation (3rd Edition). Addison-Wesley Longman Publishing Co., Inc., USA (2006)
10. van de Pol, I., van Rooij, I., Szymanik, J.: Parameterized complexity of theory of mind reasoning in dynamic epistemic logic. J of Log Lang and Inf **27**, 255–294 (2018). https://doi.org/https://doi.org/10.1007/s10849-018-9268-4