Arbitrary Public Announcement Logic with Mental Programs

Tristan Charrier
ENS Rennes, France
tristan.charrier@ens-rennes.fr

François Schwarzentruber
ENS Rennes, France
francois.schwarzentruber@ens-rennes.fr

ABSTRACT

We propose a variant of arbitrary public announcement logic which is decidable. In this variant, knowledge accessibility relations are defined by programs. Technically, programs are written in dynamic logic with propositional assignments. We prove that both the model checking problem and the satisfiability problem are decidable and $AEXP_{pol}$-complete where $AEXP_{pol}$ is the class of decision problems decided by alternating Turing machines running in exponential time where the number of alternations is polynomial. Whereas arbitrary public announcement logic is undecidable, our framework is decidable and we provide a proof-of-concept to show its expressiveness: we use our framework to reason about epistemic properties and arbitrary announcements when agents are cameras located in the plane.

Categories and Subject Descriptors

F.4.1 [**Mathematical Logic**]: Modal logic; I.2.4 [**Knowledge Representation Formalisms and Methods**]: Modal logic; I.2.11 [**Distributed Artificial Intelligence**]: Multi-agent systems

General Terms

Algorithms; Languages; Theory; Verification

Keywords

Epistemic logic; Complexity theory; Public announcement logic; Dynamic logic

1. INTRODUCTION

In a multi-robot system, agents obtain knowledge from what they perceive with their sensors and from the information in some communication channel. Dynamic epistemic logic [6][14] aims at expressing properties about the knowledge of agents and at modeling dynamics of knowledge in multi-agent settings. Public announcement logic is a fragment of full Dynamic epistemic logic. For instance, in public announcement logic, we can reason about practical situations such as robots publicly reading messages in a public channel of communication.

Appears in: *Proceedings of the 14th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2015), Bordini, Elkind, Weiss, Yolum (eds.), May, 4–8, 2015, Istanbul, Turkey.*

The truth of a construction $K_a\varphi$ (agent a knows φ) is traditionnally given by the truth of φ in all possible worlds for agent a. Here we model the set of possible worlds for agent a as the set of reachable worlds from the initial world by a given program π_a. For instance, let us consider the muddy children puzzle [14]. Each child can see the mud on others but cannot see his or her own forehead. A child imagines different possible worlds by *mentally executing* the following program:

```
(*)  non-deterministically choose $i \in \{1, 2\}$
     **if** $i = 1$ **then** clean his own forehead **else** put mud on it.
```

An interesting issue is to find out whether there exists a piece of information we can write in the public channel such that a given property φ holds, as for instance in Russian cards problem [15]. Such scenarios are captured by arbitrary public announcement [3].

The satisfiability problem in arbitrary public announcement logic (APAL) is undecidable [11] in the general settings. We propose a framework where APAL becomes decidable. Our contribution is threefold:

- We propose a framework where epistemic accessibility relations are defined by programs, as above (see (*)). Our framework combines dynamic logic of propositional assignments (DLPA) [5] for representing mental programs and arbitrary public announcement logic.
- We then prove that when knowledge is defined by mental programs, both the model checking and the satisfiability problem are $AEXP_{pol}$-complete where $AEXP_{pol}$ is the class of decision problems decided by alternating Turing machines running in exponential time where the number of alternations is polynomial. We recall that non-deterministic Turing machines allow for existential choices of transitions whereas alternating Turing machines allows for existential and universal choices of transitions during the execution [9]. The class $AEXP_{pol}$ is between EXPTIME and AEXPTIME = EXPSPACE.
- We finally provide a proof-of-concept for showing the expressiveness of the proposed logic. We use our framework to reason about arbitrary public announcements when agents are cameras located in the plane [12].

The article is organized as follows. First, in section 2, we present our framework called $\mathcal{L}_{\text{DL-PA-APAL}}$. In section 3, we present optimal $AEXP_{pol}$ procedures for the $\mathcal{L}_{\text{DL-PA-APAL}}$-model checking problem and the satisfiability problem. In section 4, we prove their $AEXP_{pol}$-hardness. In section 5, we give our proof-of-concept. Section 6 presents the related work. Finally, in section 7, we discuss our contribution and provide directions for future research.

2. OUR FRAMEWORK

2.1 Syntax

Let ATM be a countably infinite set of atomic propositions whose typical members are denoted by $p, q, p_1, p_2, \ldots$. The language of $\mathcal{L}_{\text{DL-PA-APAL}}$ is defined by the following BNF:

$$\begin{array}{lcl} \varphi & ::= & \top \mid p \mid \neg\varphi \mid (\varphi \vee \varphi) \mid \hat{K}_\pi\varphi \mid \langle\varphi!\rangle\varphi \mid \Diamond\varphi \\ \pi & ::= & p{\leftarrow}\bot \mid p{\leftarrow}\top \mid \beta? \mid \pi;\pi \mid \pi\cup\pi \end{array}$$

where $p \in ATM$ and β is a propositional formula. As usual, we introduce the following abbreviations: $(\varphi \wedge \psi) := \neg(\neg\varphi \vee \neg\psi)$, $(\varphi \rightarrow \psi) := (\neg\varphi \vee \psi)$, $(\varphi \leftrightarrow \psi) := (\varphi \rightarrow \psi) \wedge (\psi \rightarrow \varphi)$, $K_\pi\varphi := \neg\hat{K}_\pi\neg\varphi$, $[\varphi!]\psi := \neg\langle\varphi!\rangle\neg\psi$ and $\Box\varphi = \neg\Diamond\neg\varphi$. The construction $\hat{K}_\pi\varphi$ is read 'there is an execution of π such that φ holds after it'. The construction $K_\pi\varphi$ is read 'for all executions of π, φ holds', or 'all executions of π lead to a φ-world', or 'agent a knows φ where agent a uses π as the program for computing mental states'. Programs π aim at representing syntactically how agents are able to reach one of their mental states. Intuitive meanings of program constructions are given in the following table:

$p{\leftarrow}\bot$	we set p to false
$p{\leftarrow}\top$	we set p to true
$\beta?$	test whether β is true
$\pi;\pi'$	we execute π then π'
$\pi\cup\pi'$	we execute non-deterministically π or π'

For instance, the following program non-deterministically chooses values for atomic propositions in $\{p_1, ..., p_n\}$:

$$ch(p_1, ..., p_n) = (p_1{\leftarrow}\bot \cup p_1{\leftarrow}\top); \ldots; (p_n{\leftarrow}\bot \cup p_n{\leftarrow}\top).$$

Note that operators K_π are *not* dynamic operators. Even if they are indexed by a program π, the program π only specifies how one agent goes from the current world to a possible world but it does not change the model.

On the contrary, $\langle\varphi!\rangle$ and $\Diamond$ are dynamic operators. The construction $\langle\varphi!\rangle\psi$ means 'φ is true and ψ holds after the public announcement of φ'. The construction $\Diamond\varphi$ means 'there exists a true public announcement such that φ holds after the announcement'. For all formulas φ, let $ATM(\varphi)$ be the set of atomic propositions appearing in φ.

Example 1. *Let us focus on the muddy children example given in the introduction. Let p_a be the proposition denoting 'a's forehead is muddy'. The program (*) that corresponds to change the current mental state for child a is $ch(p_a)$. Thus, the construction $K_{ch(p_a)}\varphi$ is read as 'agent a knows φ'. The $\mathcal{L}_{\text{DL-PA-APAL}}$-formula*

$$\Diamond\langle(\neg K_{ch(p_a)}p_a \wedge \neg K_{ch(p_b)}p_b)!\rangle K_{ch(p_a)}p_a$$

is read 'there is a public announcement such after having announced that agent a does not know a is muddy and agent b does not know b is muddy, agent a knows a is muddy'.

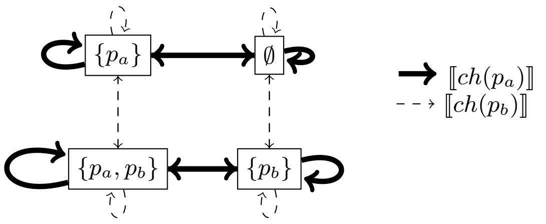


Figure 1: Set of valuations for the muddy children example with child a and child b

2.2 Semantics

Let $W_{all} = 2^{ATM}$ be the set of all valuations.

Definition 1. *Let $W \subseteq W_{all}$. We define $[\![\varphi]\!]_W$ and $[\![\pi]\!]$ by structural induction as follows:*

$$\begin{array}{ll} [\![\top]\!]_W & = W_{all}; \\ [\![p]\!]_W & = \{w \in W_{all} \mid p \in w\}; \\ [\![\neg\varphi]\!]_W & = 2^{ATM} \setminus [\![\varphi]\!]_W; \\ [\![\varphi\vee\psi]\!]_W & = [\![\varphi]\!]_W \cup [\![\psi]\!]_W; \\ [\![\hat{K}_\pi\varphi]\!]_W & = \left\{w \in W_{all} \mid \begin{array}{l}\textit{there exists } u \in W, (w,u) \in \\ [\![\pi]\!] \textit{ and } u \in [\![\varphi]\!]_W\end{array}\right\}; \\ [\![\langle\varphi!\rangle\psi]\!]_W & = [\![\varphi]\!]_W \cap [\![\psi]\!]_{W\cap[\![\varphi]\!]_W}; \\ [\![\Diamond\varphi]\!]_W & = \left\{u \in W \mid \begin{array}{l}\textit{there exists a formula } \psi \textit{ such} \\ \textit{that } u \in [\![\langle\psi!\rangle\varphi]\!]_W\end{array}\right\}; \end{array}$$

$$\begin{array}{ll} [\![p{\leftarrow}\bot]\!] & = \{(w,u) \in W_{all}^2 \mid u = w \setminus \{p\}\}; \\ [\![p{\leftarrow}\top]\!] & = \{(w,u) \in W_{all}^2 \mid u = w \cup \{p\}\}; \\ [\![\pi;\pi']\!] & = [\![\pi]\!] \circ [\![\pi']\!]; \\ [\![\pi\cup\pi']\!] & = [\![\pi]\!] \cup [\![\pi']\!]; \\ [\![\beta?]\!] & = \{(w,w) \in W_{all}^2 \mid w \models_{PL} \beta\} \end{array}$$

where $w \models_{PL} \beta$ means that the propositional formula β is true in the valuation w.

We write $W, w \models \varphi$ for all $w \in [\![\varphi]\!]_W$. W denotes the current set of valuations. After the announcement of φ, we restrict the set of possible valuations to those where φ holds. After an arbitrary announcement, we restrict the current set of valuations to an arbitrary subset U containing the current valuation.

Remark 1. *Actually, the definition of $[\![\Diamond\varphi]\!]_W$ (see Definition 1) is equivalent to the following truth condition:*

$$[\![\Diamond\varphi]\!]_W \quad = \left\{u \in W_{all} \mid \begin{array}{l}\textit{there exists } U \textit{ such that} \\ \{u\} \subseteq U \subseteq W \textit{ and } u \in [\![\varphi]\!]_U\end{array}\right\}$$

because all subsets of valuations U restricted to the finite set $ATM(\varphi)$ can been described by a formula by

$$\psi = \bigvee_{u\in U}\left(\bigwedge_{p\in u\cap ATM(\varphi)} p \wedge \bigwedge_{p\in ATM(\varphi)\setminus u} \neg p\right).$$

Example 2. *Figure 1 shows the set of valuations for the muddy children example and relations $[\![ch(p_a)]\!]$ and $[\![ch(p_b)]\!]$ that are the epistemic accessibility relations for respectively agent a and b.*

When we execute a program, we do not require that the valuation is in W at each step. According to Definition 1, we only require the valuation at the end of the execution of a program π in a construction of the form $\hat{K}_\pi \varphi$ to be in W. Indeed, when we define knowledge with programs, only the effect of the whole program matters. Intermediate valuations during the execution of a program have no specific meaning. In the example of Figure 1, after having announced that the state of heads are the same, only valuations in $\{\emptyset, \{p_a, p_b\}\}$ remain. We have $\{\emptyset, \{p_a, p_b\}\}, \{p_a, p_b\} \not\models \hat{K}_{p_a \leftarrow \bot} \hat{K}_{p_b \leftarrow \bot} \top$. But we have $\{\emptyset, \{p_a, p_b\}\}, \{p_a, p_b\} \models \hat{K}_{p_a \leftarrow \bot ; p_b \leftarrow \bot} \top$ although the valuation $\{p_b\}$ obtained after having executed $p_a \leftarrow \bot$ is not in $\{\emptyset, \{p_a, p_b\}\}$.

2.3 Decision problems

The $\mathcal{L}_{\text{DL-PA-APAL}}$-model checking problem is formally defined by:

- **Input**: a valuation w and a $\mathcal{L}_{\text{DL-PA-APAL}}$-formula φ;
- **Output**: yes if and only if $W_{all}, w \models \varphi$.

The $\mathcal{L}_{\text{DL-PA-APAL}}$-satisfiability problem is formally defined by:

- **Input**: a $\mathcal{L}_{\text{DL-PA-APAL}}$-formula φ;
- **Output**: yes if and only if there exists a valuation w such that $W_{all}, w \models \varphi$.

3. UPPER BOUNDS

We recall that $AEXP_{pol}$ stands for the class of problems computable on an alternating Turing machine running in exponential time with a polynomial number of alternations. The aim of this section is to prove the following theorems:

THEOREM 1. *The $\mathcal{L}_{\text{DL-PA-APAL}}$-model checking problem is in $AEXP_{pol}$.*

THEOREM 2. *The $\mathcal{L}_{\text{DL-PA-APAL}}$-satisfiability problem is in $AEXP_{pol}$.*

Theorem 1 will be proven after having defined the main model checking procedure Mc, and having proven its correctness (Proposition 1) and its complexity (Proposition 2).

Mc calls the sub-procedures mc_{yes} and mc_{no} described in Figure 2 and if $w \in W$, the call $mc_{yes}(W, w, \varphi)$ (respectively $mc_{no}(W, w, \varphi)$) fails if and only if $W, w \not\models \varphi$ (respectively $W, w \models \varphi$). If the call $mc_{yes}(W_{all}, w, \varphi)$ does not fail, the call $Mc(w, \varphi)$ succeeds. Alternating algorithms contain non-deterministic choices, also called existential choices (($\exists$) choices) and universal choices (($\forall$) choices).

In the case where φ is of the form $\hat{K}_\pi \psi$, the procedure mc_{yes} guesses a valuation u, then calls a subroutine $ispath_{yes}$ shown in figure 3 checking whether u is reachable from w by executing the program π and checks whether ψ holds in u. If there is no such u the subroutine fails. In the same case, the procedure mc_{no} checks whether for all $u \in W$, either u is not reachable from w by π (that is $ispath_{no}(w, u, \pi)$ does not fail) or ψ does not hold in u. Remark that in $ispath_{yes}$ and $ispath_{no}$, the model checking of propositional formulas ($w \models_{PL} \beta$) is implemented by a deterministic function in polynomial time.

In the case where φ is of the form $\Diamond\chi$, the procedure mc_{yes} guesses a subset W'' of W containing w and checks whether χ holds in w when we restrict to worlds in W''. Similarly, in the same case, the procedure mc_{no} checks that for all subsets W'' of W containing w, χ does not hold in w when we restrict to worlds in W''.

In the case where φ is of the form $\langle\psi!\rangle\chi$, the procedure mc_{yes} first check that ψ holds. Then $[\![\psi]\!]_W$ is computed

```
procedure $Mc(w, \varphi)$
    $mc_{yes}(W_{all}, w, \varphi)$
    accept
```

```
procedure $mc_{yes}(W, w, \varphi)$
    match $\varphi$ with
        case $\varphi = p$: if $p \notin w$ then reject
        case $\varphi = \neg\psi$: $mc_{no}(W, w, \psi)$
        case $\varphi = \psi_1 \vee \psi_2$:
            $(\exists)$ choose $i \in \{1, 2\}$
            $mc_{yes}(W, w, \psi_i)$
        case $\varphi = \hat{K}_\pi \psi$:
            $(\exists)$ choose $u \in W$
            $ispath_{yes}(w, u, \pi)$
            $mc_{yes}(W, u, \psi)$
        case $\varphi = \langle\psi!\rangle\chi$:
            $mc_{yes}(W, w, \psi)$
            $(\exists)$ choose $W' \subseteq W \backslash \{w\}$
            $W'' = W' \cup \{w\}$
            $(\forall)$ choose $u \in W''$
            $(\forall)$ choose $v \in W \backslash W''$
            $mc_{yes}(W, u, \psi)$
            $mc_{no}(W, v, \psi)$
            $mc_{yes}(W'', w, \chi)$
        case $\varphi = \Diamond\chi$:
            $(\exists)$ choose $W' \subseteq W \backslash \{w\}$
            $W'' = W' \cup \{w\}$
            $mc_{yes}(W'', w, \chi)$
```

```
procedure $mc_{no}(W, w, \varphi)$
    match $\varphi$ with
        case $\varphi = p$: if $p \in w$ then reject
        case $\varphi = \neg\psi$: $mc_{yes}(W, w, \psi)$
        case $\varphi = \psi_1 \vee \psi_2$:
            $(\forall)$ choose $i \in \{1, 2\}$
            $mc_{no}(W, w, \psi_i)$
        case $\varphi = \hat{K}_\pi \psi$:
            $(\forall)$ choose $u \in W$
            $(\exists)$ choose $i \in \{0, 1\}$
            if $i = 0$ then
                $ispath_{no}(w, u, \pi)$
            else
                $mc_{no}(W, u, \psi)$
        case $\varphi = \langle\psi!\rangle\chi$:
            $(\exists)$ choose $i \in \{0, 1\}$
            if $i = 0$ then
                $mc_{no}(W, w, \psi)$
            else
                $(\exists)$ choose $W' \subseteq W \backslash \{w\}$
                $W'' = W' \cup \{w\}$
                $(\forall)$ choose $u \in W''$
                $(\forall)$ choose $v \in W \backslash W''$
                $mc_{yes}(W, u, \psi)$
                $mc_{no}(W, v, \psi)$
                $mc_{no}(W'', w, \chi)$
        case $\varphi = \Diamond\chi$:
            $(\forall)$ choose $W' \subseteq W \backslash \{w\}$
            $W'' = W' \cup \{w\}$
            $mc_{no}(W'', w, \chi)$
```

Figure 2: The main model checking procedure Mc and the two dual model checking sub-procedures mc_{yes} and mc_{no}.

```
procedure $ispath_{yes}(w, u, \pi)$
    match $\pi$ with
        case $\pi = p{\leftarrow}\bot$:
            if $u \neq w \setminus \{p\}$ then reject
        case $\pi = p{\leftarrow}\top$:
            if $u \neq w \cup \{p\}$ then reject
        case $\pi = \pi_1; \pi_2$:
            $(\exists)$ choose a valuation $v \in W_{all}$.
            $ispath_{yes}(w, v, \pi_1)$
            $ispath_{yes}(v, u, \pi_2)$
        case $\pi = \pi_1 \cup \pi_2$:
            $(\exists)$ choose $i \in \{1, 2\}$
            $ispath_{yes}(w, u, \pi_i)$
        case $\pi = \beta?$:
            if $w \neq u$ or $w \not\models_{PL} \beta$
            then reject
```

```
procedure $ispath_{no}(w, u, \pi)$
    match $\pi$ with
        case $\pi = p{\leftarrow}\bot$:
            if $u = w \setminus \{p\}$ then reject
        case $\pi = p{\leftarrow}\top$:
            if $u = w \cup \{p\}$ then reject
        case $\pi = \pi_1; \pi_2$:
            $(\forall)$ choose a valuation $v \in W_{all}$.
            $(\exists)$ choose $i \in \{1, 2\}$
            if $i = 1$
              then $ispath_{no}(w, v, \pi_1)$
              else $ispath_{no}(v, u, \pi_2)$
        case $\pi = \pi_1 \cup \pi_2$:
            $(\forall)$ choose $i \in \{1, 2\}$
            $ispath_{no}(w, u, \pi_i)$
        case $\pi = \beta?$:
            if $w = u$ and $w \models_{PL} \beta$
            then reject
```

Figure 3: The two dual path searching procedures $ispath_{yes}$ **and** $ispath_{no}$**.**

and assigned to W''. More precisely, after the dashed block, the only execution that does not fail is the execution where $W'' = [\![\psi]\!]_W$. It guesses the candidate subset W'' of W containing w that should be $[\![\psi]\!]_W$. Then we actually check that W'' is actually $[\![\psi]\!]_W$. We finally check that χ holds when we restrict to worlds in W''. Similarly, in the same case, the procedure mc_{no} checks that either ψ does not hold or that $\langle\psi!\rangle\chi$ does not hold even if ψ holds. Suppose that ψ holds (corresponding to $i = 1$). Then, it guesses the candidate subset W'' of W containing w that should be $[\![\psi]\!]_W$ in the same way as in mc_{yes} (note that the existential choice in mc_{yes} for W' is not turned into a universal one and we use exactly the same instructions put in the dashed box). We finally check that χ does not hold when we restrict to worlds in W''.

PROPOSITION 1. *$Mc(w, \varphi)$ succeeds iff $W_{all}, w \models \varphi$.*

PROOF. We proceed by induction on the structure of φ by proving the following hypothesis:

$$H_{mc}(\varphi) : \begin{cases} \text{for all } W \subseteq W_{all} \text{ and } w \in W, \\ mc_{yes}(W, w, \varphi) \text{ fails if and only if } W, w \not\models \varphi. \\ mc_{no}(W, w, \varphi) \text{ fails if and only if } W, w \models \varphi. \end{cases}$$

mc_{yes} and mc_{no} use the procedures $ispath_{yes}$ and $ispath_{no}$, so we need to prove this lemma first:

LEMMA 1. *For all $w, u \in W_{all}$,*

- *$ispath_{yes}(w, u, \pi)$ fails if and only if there is no π-path from w to u;*
- *$ispath_{no}(w, u, \pi)$ fails if and only if there is a π-path from w to u.*

Details are left to the reader. □

PROPOSITION 2. *The procedure Mc is implementable by an alternating Turing machine running in exponential time and with a polynomial number of alternations.*

PROOF. To prove this proposition, we first need to analyze the complexity in time of the functions $ispath_{yes}$ and $ispath_{no}$. Both $ispath_{yes}$ and $ispath_{no}$ can be implemented by an alternating Turing machine in polynomial time, because choosing a valuation is doable in polynomial time. All remaining operations in the algorithm introduce polynomial factors. To ease the presentation, we bound their complexity by an exponential. Without loss of generality we have:

$$T_{ispath}(w, u, \pi) \leq 2^{(\#ATM+1)\times|\pi|}$$

where $T_{ispath}(w, u, \pi)$ is the time of execution for $ispath_{yes}(w, u, \pi)$ and $ispath_{no}(w, u, \pi)$. Now we prove that mc_{yes} and mc_{no} are running in exponential time by proving by induction on φ that the following proposition $H(\varphi)$, in which $T_{mc}(W, w, \varphi)$ stands for the time of execution for $mc_{yes}(W, w, \varphi)$ and $mc_{no}(W, w, \varphi)$ is true:

$$\text{for all } W \subseteq W_{all}, w \in W, T_{mc}(W, w, \varphi) \leq 2^{(\#ATM+1)\times|\varphi|}.$$

Details of the proof are left to the reader. Now in the call $Mc(w, \varphi)$, W_{all} is constructed with respect to the atomic propositions in φ, so we assume that $\#ATM \leq |\varphi|$. To sum up, the time of $Mc(w, \varphi)$ is bounded by $2^{|\varphi|^2+|\varphi|}$, so the problem of model checking $\mathcal{L}_{\text{DL-PA-APAL}}$-formulas is in AEXPTIME.

Concerning the number of alternations, we remark that each symbol in φ introduces at most 2 alternations in the execution. So the number of alternations is linear. To conclude, the problem is in $AEXP_{pol}$. □

In order to prove Theorem 2, we polynomially reduce the $\mathcal{L}_{\text{DL-PA-APAL}}$-satisfiability problem to the $\mathcal{L}_{\text{DL-PA-APAL}}$-model checking problem as follows: an instance φ of the former is translated in the instance $\emptyset, \hat{K}_{ch(ATM(\varphi))}\varphi$ of the latter.

4. LOWER BOUNDS

The aim of this section is to prove the two following theorems:

THEOREM 3. *The $\mathcal{L}_{\text{DL-PA-APAL}}$-model checking problem is $AEXP_{pol}$-hard.*

THEOREM 4. *The $\mathcal{L}_{\text{DL-PA-APAL}}$-satisfiability problem is $AEXP_{pol}$-hard.*

Note that we can polynomially reduce the $\mathcal{L}_{\text{DL-PA-APAL}}$-model checking problem to the $\mathcal{L}_{\text{DL-PA-APAL}}$-satisfiability problem as follows: an instance w, φ of the former is translated in the instance $\varphi \wedge \bigwedge_{p \in w \cap ATM(\varphi)} p \wedge \bigwedge_{p \notin ATM(\varphi) \setminus w} \neg p$ of the latter. Hence Theorem 4 is implied by theorem 3. The rest of the section is about the proof of 3.

First we tackle the lower bound of the fragment $\mathcal{L}_{1\diamond}$ of $\mathcal{L}_{\text{DL-PA-APAL}}$-formulas of the form $\diamond\varphi$ where φ does not contain any announcement. Finally we show the $AEXP_{pol}$-hardness for the whole logic.

4.1 NEXPTIME-hardness with one arbitrary announcement

PROPOSITION 3. *The $\mathcal{L}_{1\diamond}$-model checking problem is NEXPTIME-hard.*

PROOF. Let L be a NEXPTIME problem. We are going to define a polynomial reduction tr from L to the $\mathcal{L}_{1\diamond}$-model checking problem. Let M be an non-deterministic Turing machine running in exponential time that decides L. Let P be a polynomial function such that for all inputs ω, the length of any execution of M on ω is bounded by $2^{P(|\omega|)}$. Note that the length of the non-empty part of the tape is bounded also by $2^{P(|\omega|)}$ at any step of the execution. Let ω be an instance of L. We encode the existence of an accepting execution of M for ω (that is, which ends in the state *accept*) in an $\mathcal{L}_{1\diamond}$-model checking instance $tr(\omega)$. We suppose that we loop on the state *accept*. Here, we represent an execution of M in an array depicted in Figure 4a.

The integers x and t in $\left\{0, \ldots, 2^{P(|\omega|)} - 1\right\}$ respectively represent a cell index on the tape and the time index. We identify x and t with their binary representations respectively expressed by means of the atomic propositions $x_1, \ldots, x_n$ and $t_1, \ldots, t_n$ where $n = P(|\omega|)$.

We also use these atomic propositions to represent a configuration of M:

- $state_s$: M is in the state s (s could be the initial state *init*, the accepting state *accept* or any other state of the finite state space of the Turing machine);
- cur: the cursor of the tape is in position x at time t;
- $tape_a$: the symbol a is stored at position x at time t;
- $tr_{q,a,e,b,d}$: the transition starting from state q and going to state e, reading the symbol a on the tape and writing b, moving the cursor on direction d is going to be executed at time t.

The set of the above atomic propositions is noted ATM'. Remark that there are unfitting valuations in W_{all}: for instance, valuations in which both $tape_a$ and $tape_b$ for $a \neq b$ are true. The idea goes as follow. We perform an arbitrary public announcement so that the remaining valuations describe an accepting execution for ω of M. Each remaining valuation describes the content of one cell on the tape at a given time.

We define the following abbreviations:

- There is a unique true proposition among $\{p_1, ..., p_n\}$:
$$\exists!(p_1, ..., p_n) := \bigvee_{i\in\{1,\ldots,n\}} \left(p_i \wedge \bigwedge_{j\neq i} \neg p_j \right);$$
- $x{=}i$, $x{\geq}i$, etc. are formulas whose intended meaning is that the values of propositions $x_1, \ldots, x_n$ are such that the number x is equal to i, greater or equal than i, etc.
- Non-deterministically choose values for $\{p_1, ..., p_n\}$ such that at least one p_i's value changes:

$$\begin{array}{rcl} ch_{\neq}(p_1, ..., p_n) & := & ((p_1?; \overline{p_1}{\leftarrow}\top) \cup (\neg p_1?; \overline{p_1}{\leftarrow}\bot)) \ ; \\ & & \ldots \ ; \\ & & ((p_n?; \overline{p_n}{\leftarrow}\top) \cup (\neg p_n?; \overline{p_n}{\leftarrow}\bot)) \ ; \\ & & ch(p_1, ..., p_n) \ ; \\ & & \left(\neg \bigwedge_{i\in\{1,\ldots,n\}} (p_i \leftrightarrow \overline{p_i}) \right)? \end{array}$$

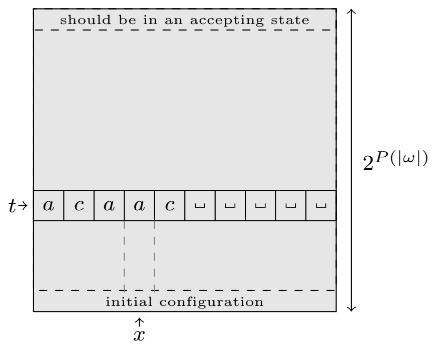
where $\overline{p_1} \ldots \overline{p_n}$ are new propositions;

- Non-deterministically change values for all atomic propositions:
$\mathscr{U} = ch(ATM')$;
- $x{\leftarrow}x{-}1$, $x{\leftarrow}x{+}1$ and $t{\leftarrow}t{+}1$ are programs of polynomial size in $|\omega|$ that respectively decrements x, increments x and increments t. We suppose that $x{\leftarrow}x{+}1$ is *not executable* when $x = 2^{P(|\omega|)} - 1$ and so on;

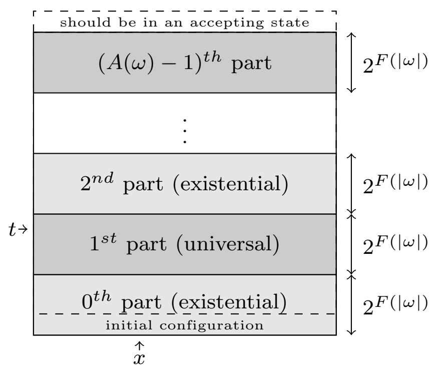
The instance $tr(\omega)$ is of the form $(w, \diamond K_{\mathscr{U}} (\varphi_{init} \wedge \varphi_{accept} \wedge \varphi_{exe}))$ where:

1. $\exists!(state_q)_{q\in Q}$(where Q is the finite set of states of M)
in each world, we have only one state
2. $\exists!(tape_a)_{a\in\Sigma}$(where Σ is the finite tape alphabet)
in each world, only one letter is written
3. $\exists!(tr_\delta)_{\delta\in\Delta}$(where Δ is the finite set of transitions of M)
in each world, only one transition is considered
4. $\hat{K}_{ch(x,tape,cur,tr)} cur$
on each possible tape, there must be a cursor
5. $cur \rightarrow K_{ch_{\neq}(x,tape,cur,tr)} \neg cur$
on each possible tape, there is only one cursor
6. $\left(state_t \rightarrow K_{ch(x,state,tape,cur,tr)} state_t\right)$
on each possible tape, only one state is considered
7. $\bigwedge_\delta \left(tr_\delta \rightarrow K_{ch(x,state,tape,cur,tr)} tr_\delta\right)$
on each possible tape, only one transition is considered
8. $\hat{K}_{x\leftarrow x+1 \cup x=2^{P(|\omega|)}-1?} \top \wedge \hat{K}_{t\leftarrow t+1 \cup t=2^{P(|\omega|)}-1?} \top$
the model contains a grid
9. $tape_a \rightarrow K_{ch(tape)} tape_a$
on each cell, there is no world in which all atomic propositions are the same except the letter written in the cell
10. $(cur \rightarrow K_{ch(cur)} cur) \wedge (\neg cur \rightarrow K_{ch(cur)} \neg cur)$
on each cell, there is no world in which all atomic propositions are the same except cur
11. $\bigwedge_a (\neg cur \wedge tape_a \rightarrow K_{t\leftarrow t+1} tape_a$
The cells where the cursor is not remain unchanged
12. $\bigwedge_{(q,a,e,b,d)\in\Delta} tr_{q,a,e,b,d} \rightarrow state_q$
The transition is compatible with the current state
13. $\bigwedge_{(q,a,e,b,d)\in\Delta} tr_{q,a,e,b,d} \wedge cur \rightarrow tape_a$
The transition is compatible with the cell under the cursor
14. $\bigwedge_{(q,a,e,b,d)\in\Delta} tr_{q,a,e,b,d} \wedge cur \rightarrow K_{t\leftarrow t+1} tape_b$
The transition is compatible with the written symbol
15. $\bigwedge_{(q,a,e,b,d)\in\Delta} tr_{q,a,e,b,d} \rightarrow K_{t\leftarrow t+1} state_e$
The transition is compatible with the next state
16. $\bigwedge_{(q,a,e,b,+1)\in\Delta} tr_{q,a,e,b,d} \wedge cur \rightarrow K_{t\leftarrow t+1; x\leftarrow x+1} cur$
Behavior of the cursor for moving to the right
17. $\bigwedge_{(q,a,e,b,-1)\in\Delta} tr_{q,a,e,b,d} \wedge cur \rightarrow K_{t\leftarrow t+1; x\leftarrow x-1} cur$
Behavior of the cursor for moving to the left
18. $\bigwedge_{(q,a,e,b,0)\in\Delta} tr_{q,a,e,b,d} \wedge cur \rightarrow K_{t\leftarrow t+1} cur$
Behavior of the cursor for staying at the same cell

Table 1: Formulas that constraint the set of valuations in order to represent an execution



a. NEXPTIME case


should be in an accepting state
$(A(\omega)-1)^{th}$ part
$2^{F(|\omega|)}$
2^{nd} part (existential)
$2^{F(|\omega|)}$
t
1^{st} part (universal)
$2^{F(|\omega|)}$
0^{th} part (existential)
$2^{F(|\omega|)}$
initial configuration
x

b. $AEXP_{pol}$ case

Figure 4: Execution of M

- w is the valuation $\{tape_{\omega_0}, cur, state_{init}\}$;
- $\varphi_{init} := t{=}0 \to [(x{=}0 \to (cur \wedge tape_{\omega_0}))\wedge \bigwedge_{i=1}^{|\omega|-1}(x{=}i \to (\neg cur \wedge tape_{\omega_i}))\wedge (x{\geq}|\omega| \to (\neg cur \wedge tape_{␣}))]$ where ␣ is the blank symbol;
- $\varphi_{accept} := \left(t{=}2^{P(|\omega|)}{-}1\right) \to state_{accept}$;
- and the formula φ_{exe} states that the resulting valuations represent an execution of M. Formula φ_{exe} is the conjunction of the formulas in the table 1.

The valuation w represents the most-left tape cell in initial configuration of M (note that $w \models (x = 0 \wedge t = 0)$). Formula φ_{exe} (see formula 8 of Table 1) imposes that the surviving valuations contain a grid (see Figure 4a) starting from w and representing an execution of M. Formula φ_{init} says that the row for $t = 0$ is the initial configuration for word ω. Formula φ_{accept} says that the row for $t = 2^{P(|\omega|)}{-}1$ is an accepting configuration.

$tr(\omega)$ is computable from ω in polynomial time in $|\omega|$ and is a positive instance of $\mathcal{L}_{1\Diamond}$-model checking problem if and only if ω is a positive instance of L.

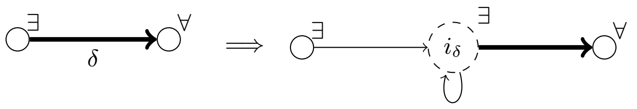
□

4.2 $AEXP_{pol}$-hardness for the full language

Now we are ready to prove Theorem 3. The proof is similar to that of proposition 3. The difference is that we will use both $\Diamond$ and $\Box$ (its dual) operators to simulate the alternation in the execution of the Turing machine.

Proof. Let L be a problem in $AEXP_{pol}$. We are going to define a polynomial reduction tr from L to the $\mathcal{L}_{\text{DL-PA-APAL}}$-model checking problem. Let M be an alternating Turing machine running in exponential time that decides L with a polynomial number of alternations. Let A be a polynomial function such that for all inputs ω, the number of alternations of any execution starting from ω is bounded by $A(|\omega|)$. Without loss of generality, we suppose that the number of alternations is exactly $A(|\omega|)$ and that the initial state is existential. The execution is segmented into maximal parts in which each configuration is of the same type: either all states are existential states or all are universal states. Let F be a polynomial function such that for all inputs ω, the length of any such part is bounded by $2^{F(|\omega|)}$.

We transform the machine M so that the length of any part of an execution is exactly $2^{F(|\omega|)}$ in the following way. For all transitions δ from an existential state to a universal state, we add an intermediate existential state i_δ as pictured below:



We do the same transformation for transitions that go from a universal state to an existential state. Figure 4b shows the shape of an execution of M.

Now we use the notation of the proof of proposition 3 but now numbers x and t ranges over $\left\{0,\ldots,A(|\omega|){\times}2^{F(|\omega|)}\right\}$ so we take the number n of digits of the numbers x and t to be $F(\omega)+\log_2(A(\omega))+1$. Now we define formula $\varphi^{\exists}_{exe}$ which is similar to formula φ_{exe} (see proof of theorem 3) except that we impose end states of transitions to be existential. We define formula $\varphi^{\forall}_{exe}$ which is similar to formula φ_{exe} except that we impose end states of transitions to be universal.

We define the following abbreviations:

- Go to the k^{th} part:
  $\mathscr{U}_k := \mathscr{U} ; \left(\left(k{\times}2^{F(|\omega|)}{\leq}t\right) \wedge \left(t{<}(k{+}1){\times}2^{F(|\omega|)}\right)\right)?$
- Go after the k^{th} part:
  $\mathscr{U}^{\text{after}}_k := \mathscr{U} ; \left(t{>}k{\times}2^{F(|\omega|)}\right)?$
- A formula saying that all valuations concerning the k^{th} part and beyond exist (i.e. the parts from the k^{th} part are unfixed):
  $$\varphi^k_{\text{unfixed}} := K_{\mathscr{U}^{\text{after}}_k}\left(\begin{array}{l} \hat{K}_{x\leftarrow x+1\cup\left(x=A(|\omega|)\times 2^{F(|\omega|)}\right)?}\top\wedge \\ \hat{K}_{t\leftarrow t+1\cup t=A(|\omega|)\times 2^{F(|\omega|)}?}\top\wedge \\ \bigwedge_{p\in ATM'\setminus\{x_1,\ldots,x_n,t_1,\ldots,t_n\}} \\ \left(\begin{array}{l}(p \to K_{ch(p)}p)\wedge \\ (\neg p \to K_{ch(p)}\neg p)\end{array}\right)\end{array}\right)$$
- Transitions at time $(k+1){\times}2^{F(|\omega|)}{-}1$ only end into universal states:
  $\varphi^{\exists\to\forall}_{exe,k} := \left(t{=}(k+1){\times}2^{F(|\omega|)}{-}1\right) \to \exists!(tr_\delta)_{\delta\in\Delta|\delta\text{ ends in a universal state}}$
- Transitions at time $(k+1){\times}2^{F(|\omega|)}{-}1$ only end into existential states:
  $\varphi^{\forall\to\exists}_{exe,k} := \left(t{=}(k+1){\times}2^{F(|\omega|)}{-}1\right) \to \exists!(tr_\delta)_{\delta\in\Delta|\delta\text{ ends in a existential state}}$

Now we define a sequence of formulas $(\psi_k)_{k\in\{0,\ldots,A(|\omega|)\}}$ that describe the execution of the Turing machine M one part after another. Assuming the execution is already defined until the time $2k \times 2^{F(|\omega|)}$, the formula ψ_{2k} chooses the next transitions to execute in the $(2k)^{th}$ part so that the last transition executed in the $(2k)^{th}$ part leads to a universal state. Moreover, it leaves all valuations in remaining parts unfixed so that formula ψ_{2k+1} is carrying on the rest of the execution. Assuming the execution is already defined until the time $(2k+1) \times 2^{F(|\omega|)}$, the formula ψ_{2k+1} checks that all possible executions in the $(2k+1)^{th}$ part are accepting. Assuming the execution is defined until the end, the last formula $\psi_{A(|\omega|)}$ checks that the last state is *accept*. Formally:

- $\psi_{A(|\omega|)} := K_{\mathscr{U}_{A(|\omega|)}}[(t = A(|\omega|) \times 2^{F(|\omega|)}) \rightarrow state_{accept}]$;
- for all k such that $2k+1 < A(|\omega|)$, $\psi_{2k+1} := \Box \left(\left(K_{\mathscr{U}_{2k+1}}(\varphi^{\forall}_{exe} \wedge \varphi^{\forall \rightarrow \exists}_{exe,2k+1}) \wedge \varphi^{2k+2}_{\text{unfixed}} \right) \rightarrow \psi_{2k+2} \right)$;
- for all k such that $0 \leq 2k < A(|\omega|)$, $\psi_{2k} := \Diamond \left(K_{\mathscr{U}_{2k}}(\varphi^{\exists}_{exe} \wedge \varphi^{\exists \rightarrow \forall}_{exe,2k}) \wedge \varphi^{2k+1}_{\text{unfixed}} \wedge \psi_{2k+1} \right)$;

For all $k \in \{0, \ldots, A(|\omega|)\}$, we define the following property $P(k)$:

For all $W \subseteq W_{all}$, such that W defines a unique configuration c at time $t = k \times 2^{F(|\omega|)}$ and contains all valuations for $t > k \times 2^{F(|\omega|)}$, $W, w \models \psi_k$ if and only if c is an accepting configuration.

We can prove by induction on k that $P(k)$ is true for all $k \in \{0, \ldots, A(|\omega|)\}$.

The instance $tr(\omega)$ is of the form (w, φ) where:

- w is the valuation $\{tape_{\omega_0}, cur, state_{init}\}$;
- $\varphi = \Diamond \left(K_{\mathscr{U}} \varphi_{init} \wedge \varphi^{0}_{\text{unfixed}} \wedge \psi_0 \right)$.

The formula φ ensures that the initial configuration ($t = 0$) is fixed, leaves all valuations concerning $t > 0$ unfixed and ψ_0 checks whether the initial configuration is accepting. $tr(\omega)$ is computable from ω in polynomial time in $|\omega|$. We have that ω is a positive instance of L if and only if the initial configuration for ω in M is accepting if and only if (w, φ) is a positive instance of the $\mathcal{L}_{\text{DL-PA-APAL}}$-model checking problem. Hence the $\mathcal{L}_{\text{DL-PA-APAL}}$-model checking problem is $AEXP_{pol}$-hard. $\Box$

5. PROOF-OF-CONCEPT: AUTONOMOUS CAMERAS

We are interested here in general scenarios involving epistemic properties, public announcements and arbitrary public announcements where agents are autonomous cameras that see cones, as depicted in figure 5. The framework defined in [12], called Big brother Logic, enables reasoning about epistemic properties when agents are cameras. First we extend Big Brother Logic with public announcements and arbitrary announcements. Secondly, we want to show that reasoning in this extension is decidable. It is not relevant to embed it into the abstract version of arbitrary public announcement logic since it is undecidable [11]. Instead, we show how to embed the extension into $\mathcal{L}_{\text{DL-PA-APAL}}$ in order to obtain decidability.

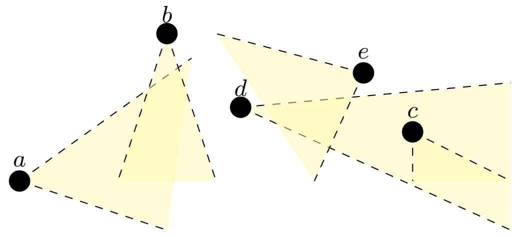


Figure 5: Five cameras and their sectors of vision

5.1 Extension with public and arbitrary announcements

We extend the language introduced in [12] and [4] with public announcement and arbitrary public announcement operators:

$$(\mathcal{L}_{\text{BBL-APAL}}) \quad \varphi \ ::= \ a \triangleright b \ \mid \ \neg\varphi \mid \varphi \vee \varphi \mid \mathsf{K}_{\mathsf{a}}\varphi \mid \langle\varphi!\rangle\varphi \mid \Diamond\varphi$$

where a, b are agents. The atomic proposition $\mathsf{a} \triangleright \mathsf{b}$ is read "agent a sees agent b". The construction $\mathsf{K}_{\mathsf{a}}\varphi$ is read "agent a knows φ" and $\langle\varphi!\rangle\psi$ and $\Diamond\varphi$ are respectively the constructions of public announcement and arbitrary announcement.

In [12], the semantics is given in term of models where each agent has a position, an angle of cone and a direction of view and there is common knowledge of the positions and the angles of cones. The authors give also an abstraction that goes as follows. For all agents a, the set S_a denotes the possible *vision sets*, that are the sets of agents that a may see. From positions and angles of cones, we can compute sets of vision sets $(S_{\mathsf{a}})_{\mathsf{a} \in AGT}$ in polynomial time in the number of agents. That is, the collection $(S_{\mathsf{a}})_{\mathsf{a} \in AGT}$ gives all the abstract information for reasoning about what agents see and know.

EXAMPLE 3. *In figure 5 we have (trust us about the angles):*

- $S_{\mathsf{a}} = \{\emptyset, \{\mathsf{b}\}, \{\mathsf{b}, \mathsf{d}\}, \{\mathsf{b}, \mathsf{d}, \mathsf{e}\}, \{\mathsf{d}, \mathsf{e}, \mathsf{c}\}, \{\mathsf{e}, \mathsf{c}\}, \{\mathsf{c}\}\}$
  (that is, a *can see nobody, can see only* b*, can see only* b *and* d*, etc., depending on a's direction)*
- $S_{\mathsf{b}} = \{\emptyset, \{\mathsf{a}\}, \{\mathsf{e}\}, \{\mathsf{e}, \mathsf{c}\}, \{\mathsf{c}, \mathsf{d}\}, \{\mathsf{d}\}\}$.

Now, given vision sets $(S_{\mathsf{a}})_{\mathsf{a} \in AGT}$, we define a Kripke model.

DEFINITION 2. *Given $(S_{\mathsf{a}})_{\mathsf{a} \in AGT}$, a vision-based abstract model is a triple $\mathcal{M} = (W, \sim, \sigma)$ where:*

- $W = \{(\Gamma_{\mathsf{a}})_{\mathsf{a} \in \mathsf{Agt}} \mid$ *for all* $\mathsf{a} \in \mathsf{Agt}, \Gamma_{\mathsf{a}} \in S_{\mathsf{a}}\}$*;*
- $\sim$ *maps each agent* a *to the equivalence relation over* W *defined by:*
  $(\Gamma_{\mathsf{a}})_{\mathsf{a} \in \mathsf{Agt}} \sim_{\mathsf{a}} (\Gamma'_{\mathsf{a}})_{\mathsf{a} \in \mathsf{Agt}}$ *iff* $\Gamma_{\mathsf{a}} = \Gamma'_{\mathsf{a}}$ *and for all* $\mathsf{b} \in \Gamma_{\mathsf{a}}$, $\Gamma_{\mathsf{b}} = \Gamma'_{\mathsf{b}}$*;*
- *for all* $(\Gamma_{\mathsf{a}})_{\mathsf{a} \in \mathsf{Agt}} \in W$, $(\Gamma_{\mathsf{a}})_{\mathsf{a} \in \mathsf{Agt}} \in \sigma(\mathsf{b} \triangleright \mathsf{c})$ *iff* $c \in \Gamma_{\mathsf{b}}$.

In the Kripke model, W is the set of possible worlds and a world is a function which map each agent a to a vision set Γ_a of S_a. For instance, figure 5 depicts the world where $\Gamma_a = \{c, d, e\}, \Gamma_b = \emptyset, \Gamma_c = \emptyset, \Gamma_d = \{c\}, \Gamma_e = \{b, d, a\}$.

The relation $\sim_{\mathsf{a}}$ is the indistinguishability relation for agent a: two words $w = (\Gamma_{\mathsf{a}})_{\mathsf{a}\in\mathsf{Agt}}$ and $u = (\Gamma'_{\mathsf{a}})_{\mathsf{a}\in\mathsf{Agt}}$ are $\sim_{\mathsf{a}}$-equivalent, if and only if agent a sees the same thing and agents seen by a see the same thing in both words. σ is the valuation. Now we consider updated models of $\mathcal{M}$. Given $\emptyset \subsetneq U \subseteq W$, the model $\mathcal{M}^U$ is defined as $(U, \sim_{|U}, \sigma_{|U})$ where ${\sim_{|U}}_{\mathsf{a}} = \ \sim_{\mathsf{a}} \cap U \times U$ and $\sigma_{|U}$ is such that for all $\mathsf{b}, \mathsf{c} \in AGT$, $\sigma_{|U}(\mathsf{b}\rhd\mathsf{c}) = \sigma(\mathsf{b}\rhd\mathsf{c}) \cap U$. The truth conditions are defined as follows:

- $\mathcal{M}^U, w \models \mathsf{b}\rhd\mathsf{c}$ iff $w \in \sigma_{|U}(\mathsf{b}\rhd\mathsf{c})$;
- $\mathcal{M}^U, w \models \mathsf{K}_{\mathsf{a}}\varphi$ iff for all $u \in {\sim_{|U}}_{\mathsf{a}}(w)$, $\mathcal{M}^U, u \models \varphi$;
- $\mathcal{M}^U, w \models \langle\varphi!\rangle\psi$ iff $\mathcal{M}^U, w \models \varphi$ and $\mathcal{M}^{\{u\in U \mid \mathcal{M}^U, u\models\varphi\}}, w \models \psi$;
- $\mathcal{M}^U, w \models \Diamond\varphi$ iff there exists $\{w\} \subseteq U' \subseteq U$ such that $\mathcal{M}^{U'}, w \models \varphi$.

5.2 Embedding into $\mathcal{L}_{\text{DL-PA-APAL}}$

In this subsection we restrict ourselves to a finite set of agents AGT that contains all the agents of the formulas we want to translate into $\mathcal{L}_{\text{DL-PA-APAL}}$. Let $S = (S_{\mathsf{a}})_{\mathsf{a}\in AGT}$ be a set of possible vision sets for all agents. Let φ_S be the following formula:

$$\bigwedge_{\mathsf{a}\in AGT} \bigvee_{\Gamma_{\mathsf{a}}\in S_{\mathsf{a}}} \left(\bigwedge_{\mathsf{b}\in\Gamma_{\mathsf{a}}} \mathsf{a}\rhd\mathsf{b} \wedge \bigwedge_{\mathsf{b}\notin\Gamma_{\mathsf{a}}} \neg\mathsf{a}\rhd\mathsf{b} \right)$$

This formula states that for all agents a, there is a vision set Γ_a and a sees agents Γ_a and only these ones. When φ_S is announced, the vision sets become common knowledge. Now, we can obtain a $\sim_{\mathsf{a}}$-equivalent world $(\Gamma'_{\mathsf{a}})_{\mathsf{a}\in\mathsf{Agt}}$ from $(\Gamma_{\mathsf{a}})_{\mathsf{a}\in\mathsf{Agt}}$ by changing what agents that are not seen by a see. This operation is performed by the following DL-PA program:

$\pi_{\mathsf{a}} = (\mathsf{a}\rhd\mathsf{b}_1? \cup (\neg\mathsf{a}\rhd\mathsf{b}_1?; ch(\mathsf{b}_1\rhd\mathsf{a}, \mathsf{b}_1\rhd\mathsf{b}_1, \ldots, \mathsf{b}_1\rhd\mathsf{b}_n)))\,; ..$
$\quad ...; (\mathsf{a}\rhd\mathsf{b}_n? \cup (\neg\mathsf{a}\rhd\mathsf{b}_n?; ch(\mathsf{b}_n\rhd\mathsf{a}, \mathsf{b}_n\rhd\mathsf{b}_1, \ldots, \mathsf{b}_n\rhd\mathsf{b}_n)))$
where $\{\mathsf{b}_1, \ldots, \mathsf{b}_n\} = AGT \setminus \{\mathsf{a}\}$.

We then define the translation tr by:

- $tr(\mathsf{b}\rhd\mathsf{c}) = \mathsf{b}\rhd\mathsf{c}$;
- $tr(\mathsf{K}_{\mathsf{a}}\psi) = K_{\pi_{\mathsf{a}}} tr(\psi)$;
- $tr(\langle\psi!\rangle\chi) = \langle tr(\psi)!\rangle tr(\chi)$;
- $tr(\Diamond\psi) = \Diamond tr(\psi)$.

Theorem 5. *Let $\mathcal{M}$ be the vision-based abstract model corresponding to S. Let φ be a $\mathcal{L}_{\text{BBL-APAL}}$-formula. We have:*

$$\mathcal{M}, (\Gamma_{\mathsf{a}})_{\mathsf{a}\in\mathsf{Agt}} \models \varphi \textit{ if and only if } W_{all}, \{\mathsf{b}\rhd\mathsf{c} \mid \mathsf{c} \in \Gamma_{\mathsf{b}}\} \models \langle\varphi_S!\rangle tr(\varphi).$$

Details of the proof are left to the reader. Theorem 5 shows a polynomial reduction from the $\mathcal{L}_{\text{BBL-APAL}}$-model checking to the $\mathcal{L}_{\text{DL-PA-APAL}}$-model checking problem.

Hence:

Corollary 1. *Both the $\mathcal{L}_{\text{BBL-APAL}}$-model checking and the $\mathcal{L}_{\text{BBL-APAL}}$-satisfiability problem are in $AEXP_{pol}$.*

6. RELATED WORK

The idea of executing programs for accessing possible worlds has already been sketched in [3], [4] and [12] but we here provide a more general framework. In order to syntactically express such programs, we propose to use DL-PA that originates from theoretical computer science as a variant of Propositional Dynamic Logic PDL with concrete programs. DL-PA has assignments of truth values to propositional variables instead of the abstract atomic programs of PDL [10]. Some of the theoretical properties of DL-PA were investigated recently [5]. In particular, it is known that both the model checking and the satisfiability problem in DL-PA without the Kleene star are PSPACE-complete.

The reader should be aware that an extension of DL-PA by epistemic operators [16] was already considered in the field of dynamic epistemic logics [17] where DL-PA are ontic actions. We insist on the fact that in this paper, DL-PA-programs are not ontic actions but *mental actions* for accessing different mental states. That is why we use the notation K_π instead of the traditional $[\pi]$ in dynamic logic. For instance, the operator $[p{\leftarrow}\bot]$ in [16] modifies publicly the real world so that p is false whereas here, $K_{p\leftarrow\bot}$ does not modify the real world: $p{\leftarrow}\bot$ is the program that is used by an agent to go from the actual world to a possible world without changing the actual world.

Contrary to the abstract case of arbitrary public announcement logic which is undecidable [11], here knowledge is defined via grounded programs π. The shape of the model is determined by the formula we want to satisfy, that is why we obtain decidability and $AEXP_{pol}$-completeness of the satisfiability problem. The reader may refer to [13], [7] and [8] for more information about $AEXP_{pol}$.

7. CONCLUSION

In this article, we propose a variant of arbitrary public announcement logic, $\mathcal{L}_{\text{DL-PA-APAL}}$, where knowledge is represented by *mental programs*. Contrary to standard arbitrary public announcement logic whose satisfiability problem is undecidable, our framework is proven to be decidable. We claim that reasoning about arbitrary public announcements can be done in practice. Interestingly, as a proof-of-concept, we show that when we extend the framework for reasoning about epistemic properties over cameras proposed in [12] with arbitrary public announcement logic , we still have a decidable logic.

We proved that both the $\mathcal{L}_{\text{DL-PA-APAL}}$-model checking problem and the $\mathcal{L}_{\text{DL-PA-APAL}}$-satisfiability problem are $AEXP_{pol}$-complete (theorems 1, 2, 3 and 4). Future work may concern arbitrary group announcements where public announcements are restricted to what agents of a group J actually know. It has recently been proven that the satisfiability problem is undecidable [1]. We conjecture that we can adapt algorithms of section 3 to the case of arbitrary group announcements when knowledge is represented by DL-PA programs. We conjecture that the upper bound remains $AEXP_{pol}$ for arbitrary group announcement logic.

Recently, epistemic planning has been proved to be undecidable in the general case [2]. It would be interested to investigate the decidability of epistemic planning when knowledge is defined by mental programs.

REFERENCES

[1] T. Agotnes, H. van Ditmarsch, and T. French. The undecidability of group announcements. In *Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems*, pages 893–900. International Foundation for Autonomous Agents and Multiagent Systems, 2014.

[2] G. Aucher and T. Bolander. Undecidability in epistemic planning. In *IJCAI 2013, Proceedings of the 23rd International Joint Conference on Artificial Intelligence, Beijing, China, August 3-9, 2013*, 2013.

[3] P. Balbiani, A. Baltag, H. P. van Ditmarsch, A. Herzig, T. Hoshi, and T. D. Lima. What can we achieve by arbitrary announcements?: A dynamic take on fitch's knowability. In D. Samet, editor, *TARK*, pages 42–51, 2007.

[4] P. Balbiani, O. Gasquet, and F. Schwarzentruber. Agents that look at one another. *Logic Journal of IGPL*, 21(3):438–467, 2013.

[5] P. Balbiani, A. Herzig, and N. Troquard. Dynamic logic of propositional assignments: A well-behaved variant of pdl. In *LICS*, pages 143–152. IEEE Computer Society, 2013.

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

[7] L. Bozzelli, H. P. van Ditmarsch, and S. Pinchinat. The complexity of one-agent refinement modal logic. In L. F. del Cerro, A. Herzig, and J. Mengin, editors, *JELIA*, volume 7519 of *Lecture Notes in Computer Science*, pages 120–133. Springer, 2012.

[8] L. Bozzelli, H. P. van Ditmarsch, and S. Pinchinat. The complexity of one-agent refinement modal logic. In F. Rossi, editor, *IJCAI*. IJCAI/AAAI, 2013.

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

[10] M. J. Fischer and R. E. Ladner. Propositional dynamic logic of regular programs. *J. Comput. Syst. Sci.*, 18(2):194–211, 1979.

[11] T. French and H. P. van Ditmarsch. Undecidability for arbitrary public announcement logic. In C. Areces and R. Goldblatt, editors, *Advances in Modal Logic*, pages 23–42. College Publications, 2008.

[12] O. Gasquet, V. Goranko, and F. Schwarzentruber. Big brother logic: Logical modeling and reasoning about agents equipped with surveillance cameras in the plane. In *Proceedings of the 2014 International Conference on Autonomous Agents and Multi-Agent Systems*, pages 325–332. International Foundation for Autonomous Agents and Multiagent Systems, 2014.

[13] D. S. Johnson. A catalog of complexity classes. In *Handbook of Theoretical Computer Science, Volume A: Algorithms and Complexity (A)*, pages 67–161. Elsevier, 1990.

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

[15] H. P. van Ditmarsch. The russian cards problem. *Studia Logica*, 75(1):31–62, 2003.

[16] H. P. van Ditmarsch, A. Herzig, and T. D. Lima. Public announcements, public assignments and the complexity of their logic. *Journal of Applied Non-Classical Logics*, 22(3):249–273, 2012.

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