Regular Model Checking Approach to Knowledge Reasoning over Parameterized Systems

Daniel Stan
Technical University of Kaiserslautern
Germany
stan@cs.uni-kl.de

Anthony W. Lin
Technical University of Kaiserslautern, MPI-SWS
Germany
lin@cs.uni-kl.de

ABSTRACT

We present a general framework for modelling and verifying epistemic properties over parameterized multi-agent systems that communicate by truthful public announcements. In our framework, the number of agents or the amount of certain resources are parameterized (i.e. not known a priori), and the corresponding verification problem asks whether a given epistemic property is true regardless of the instantiation of the parameters. For example, in a muddy children puzzle, one could ask whether each child will eventually find out whether (s)he is muddy, regardless of the number of children. Our framework is regular model checking (RMC) -based, wherein synchronous finite-state automata (equivalently, monadic second-order logic over words) are used to specify the systems. We propose an extension of public announcement logic as specification language. Of special interests is the addition of the so-called iterated public announcement operators, which are crucial for reasoning about knowledge in parameterized systems. Although the operators make the model checking problem undecidable, we show that this becomes decidable when an appropriate "disappearance relation" is given. Further, we show how Angluin's L*-algorithm for learning finite automata can be applied to find a disappearance relation, which is guaranteed to terminate if it is regular. We have implemented the algorithm and apply this to such examples as the Muddy Children Puzzle, the Russian Card Problem, and Large Number Challenge.

KEYWORDS

Epistemic; Public Announcement Logic; Regular Model Checking; Automaton Learning; Parameterized; Muddy Children

ACM Reference Format:
Daniel Stan and Anthony W. Lin. 2021. Regular Model Checking Approach to Knowledge Reasoning over Parameterized Systems. In *Proc. of the 20th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2021), Online, May 3–7, 2021*, IFAAMAS, 9 pages.

1 INTRODUCTION

Consider the standard problem of muddy children puzzle in knowledge reasoning [13]. Suppose that there are a total of N children, where $M \in \{1, \ldots, N\}$ of them has a mud on their forehead. Each child can observe whether another child (but not himself) has a mud on their forehead. The muddy children protocol goes in rounds. At each round, the father declares that there is a muddy child (i.e. with a mud on their forehead), and asks the children whether they know if they are muddy, to which the children can answer yes/no. The announcements made by the children are observable by other children. After a few rounds (more precisely M rounds), all children will discover the so-called common knowledge of which children (including themselves) are muddy and which are not, regardless of the value of the parameters M and N (e.g. see [13]).

The muddy children puzzle can be constructed as a typical example of a *parameterized verification problem* [5, 9, 18, 19] but with respect to epistemic properties. Even though the problem was shown undecidable for a simple safety property by Apt and Kozen in the 80s [7], the past twenty years or so have witnessed significant progress in parameterized verification (e.g., see [1, 9, 24, 45, 46] for surveys). Researchers resort to either (1) semi-algorithmic techniques that are applicable to general systems, but either without a termination guarantee or the method might terminate with a "don't know" answer, or (2) restriction to decidable subproblems (e.g. obtained by imposing certain structures on the parameterized systems). More recently, parameterized verification problem was also considered in the setting of multi-agent systems (e.g., see [5, 9, 18, 19, 25]). Despite this, very little work has been done on parameterized verification problem with respect to epistemic properties, which is in particular applicable in the simple setting of the muddy children example. This is an extremely challenging problem, while most research focus in parameterized system verification for a few decades has been on simple safety properties (e.g. [1, 2, 12, 29, 45]) and only recently on liveness properties (e.g. [21, 23]).

Summary of Results. We propose a framework for modelling and model-checking epistemic properties over parameterized multi-agent systems. Our emphasis in this paper is on *general semi-algorithmic solutions* that can lend themselves to automatically solve a variety of interesting examples in knowledge reasoning. While our semi-algorithm is not guaranteed to terminate in general, we provide *a general termination condition*, which is proved to subsume examples like Muddy Children Puzzle, Large Number Challenge, and Russian Card Problem. We detail our results below.

Firstly, let us recall a standard setting in the finite non-parameterized case using *Public Announcement Logic (PAL)* [31, 42] (also see [39, 41, 43], which provide more detailed modelling and a finite-state model checker). The system is represented by a finite Kripke structure, each of whose (binary) accessibility relation $\overset{a}{\rightsquigarrow}$ (for each agent a) satisfying the S5 axioms, i.e., $\overset{a}{\rightsquigarrow}$ is an equivalence relation (reflexive, symmetric, and transitive). That way, $\overset{a}{\rightsquigarrow}$ can be interpreted as knowledge-indistinguishability by agent a. PAL then is simply a standard modal logic with one accessibility relation per agent, as well as public announcement modalities $\{\varphi!\}$, whereby *each* agent learns about φ. A standard application of the public

Proc. of the 20th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2021), U. Endriss, A. Nowé, F. Dignum, A. Lomuscio (eds.), May 3–7, 2021, Online.

announcement operator is to model the announcement of a child in the muddy children protocol, who declares that he knows whether he has a mud on his forehead.

To extend the framework to the parameterized setting, there are a few problems. Firstly, since the Kripke Structure is now infinite (i.e. the union of all possible instantiations of the parameter), how do we *symbolically represent* the Kripke Structure? Secondly, a closer look at the solution to the muddy children example via PAL (or similar logics) [13, 31, 42] suggests that the formula in the logic is *different* for different numbers of muddy children. For parameterized verification, it is essential that we have a *uniform* specification for the epistemic property regardless of the instantiation of the parameters. We note that generalizations of epistemic logics that can provide such a uniform specification exist (e.g., quantified epistemic logic [8], iterated public announcement [15, 27]). However, the resulting logics are not only undecidable, but there are also no known semi-algorithmic solutions that would work for interesting examples.

Our framework (see §3) is in the spirit of *regular model checking* [1, 3, 10, 11, 37], wherein a configuration in the (parameterized) systems is represented by a string over some finite alphabet Σ, while a binary relation $\rightsquigarrow \subseteq \Sigma^* \times \Sigma^*$ is represented by an automata over the product alphabet $\Sigma \times \Sigma$. [The reader could understand a product alphabet just like a normal alphabet, where an automaton would synchronously read a pair (a, b) of symbols at each step.] The resulting Kripke structures are called *automatic Kripke structures* [10, 11, 37]. One benefit of this framework is that one could encode an infinite number of accessibility relation $\{\overset{i}{\rightsquigarrow}\}_{i\in\mathbb{N}}$ (one for each agent indexed $i = 0, 1, 2, \ldots$), where $\overset{i}{\rightsquigarrow} \subseteq \Sigma^* \times \Sigma^*$, as *one single automaton* representing $\rightsquigarrow \subseteq \Sigma^* \times \mathbb{N} \times \Sigma^*$. Since a string encoding $s(i)$ of each number $i \in \mathbb{N}$ could be given (e.g. $i = 3$ could be represented in unary), this automata could run over some product alphabet, e.g., $\Sigma \times \{0, 1\} \times \Sigma$. Second, to reason about knowledge over automatic Kripke Structures, it is important to enrich PAL with a few new features: (1) basic string reasoning (e.g. whether b occurs at an even position in the string), since configurations in the Kripke models are represented as strings (2) iterated public announcement operator $\{\varphi!\}^*$, since in general an unbounded number of public announcements need to be made in parameterized systems (e.g. one announcement per child/round in the muddy children protocol).

Our key results is as follows. First, in the absence of the iterated public announcement operators in the input formula, the model checking problem in our framework is *decidable with a nonelementary complexity* (see §4). Despite the high complexity, we show that our implementation [34] works well on examples like the parameterized version of the Russian Card Problem [40, 43] (where the total number of cards is not fixed a priori), where the tool verifies anonymous communication between two parties of the system could be achieved (see §6). Second, in the presence of the iterated public announcement operators in the input formula, although the model checking problem is in general undecidable (see §4), we provide a semi-algorithm for the problem tapping into Angluin's L* automata learning algorithm [6, 17] (see §5). To the best of our knowledge, this is the first application of automata learning methods to the parameterized model checking of epistemic properties. Loosely speaking, the learning algorithm will attempt the computation of the so-called "disappearance relation", that captures the order in which states are discarded during the announcements and is likely to exhibit regular patterns of the system. A termination guarantee is provided in this case (i.e. when the order can be represented by regular languages). We implemented the method and show that it can successfully verify the parameterized versions of the Muddy Children Protocol and the Large Number Challenge (see §6).

We refer the reader to our technical report [35] when complete proofs are omitted.

2 PRELIMINARIES

We denote $\mathbb{N}$ the set of natural numbers, and for any $n \in \mathbb{N}$, $[n] = \{x \in \mathbb{N} \mid 0 \leq x < n\}$.

Automata Background: An *alphabet* is a finite set Σ. A *word* w over Σ is a finite sequence $x_0 \ldots x_{n-1} \in \Sigma^n$, of *letters* of Σ, for some length n, which is is denoted $|w| = n$. We write $w[i] = x_i$ for its i-th letter ($i \in [|w|]$) and ϵ for the empty word of length 0.

A set of words L is called a *language*. It is *regular* if it it can be recognized by a regular expression, or equivalently by a nondeterministic automaton (NFA, e.g. see [33]). We denote $\mathrm{Reg}(\Sigma)$ the class of regular languages over Σ and recall the class is closed under concatenation, boolean operations, and Kleene star.

Let $\Sigma_1 \subseteq \Sigma_1'$ and Σ_2. Regular languages are also preserved by *Synchronous product* and *morphism*:

For $w_1 \in \Sigma_1^l$ and $w_2 \in \Sigma_2^l$ two words of the same length $l \in \mathbb{N}$, we write $w_1 \otimes w_2$ for the *synchronous product* word $w \in (\Sigma_1 \times \Sigma_2)^l$ such that $\forall i \in [l]$, $w[i] = (w_1[i], w_2[i])$. We extend $\otimes$ to languages, by defining, for any $L_1 \subseteq \Sigma_1^*$ and $L_2 \subseteq \Sigma_2^*$, $L_1 \otimes L_2 = \left\{ w_1 \otimes w_2 \;\middle|\; w_1 \in L_1 \wedge w_2 \in L_2 \cap \Sigma_2^{|w_1|} \right\}$

A *morphism* is any function $f : \Sigma_1 \to \Sigma_2$, we extend f to words over Σ_1 by defining, for any $w \in \Sigma_1^*$, $f(w) = f(w[0]) \ldots$ f(w[|w|-1]) $\in \Sigma_2^*$, then to languages over the superset Σ_1': for any $L \subseteq (\Sigma_1')^*$, $f(L) = \{f(w) \mid w \in L \cap (\Sigma_1)^*\}$.

For example, synchronous product's counterparts can be defined as the morphisms $\pi_{(\Sigma_1,-)}$ and $\pi_{(-,\Sigma_2)}$, *projections* on the first and second component, respectively.

We encode positions inside a word with the alphabet $\mathbb{B} = \{0, 1\}$ and for $0 \leq i < l$, $V(i, l) = 0^i 1 0^{l-i-1} \in \mathbb{B}^l$ encodes the i-th position.

When the meaning is clear, we will at times identify a finite automaton $\mathcal{A}$ and its recognized language $\mathcal{L}(\mathcal{A}) \in \mathrm{Reg}(\Sigma)$. In particular, whenever we claim a language L is regular, a recognizing automaton may be provided instead. Whenever $\Sigma = \Sigma_1 \times \Sigma_2$, the automaton may also be called a *length-preserving transducer*, or simply "*transducer*", as it can be interpreted as an automaton mapping a word $w_1 \in \Sigma_1^*$ to (non-deterministically) a word $w_2 \in \Sigma_2^*$ of the same length, such that $w_1 \otimes w_2 \in L$.

3 OUR FRAMEWORK

In this section, we provide our regular model checking framework to knowledge reasoning over parameterized systems. The section has two parts. First, an extension of PAL called PPAL (Parameterized PAL) that is interpreted over a parameterized Kripke structure. Second, a regular presentation of parameterized Kripke structure, over which PPAL-model checking is decidable.

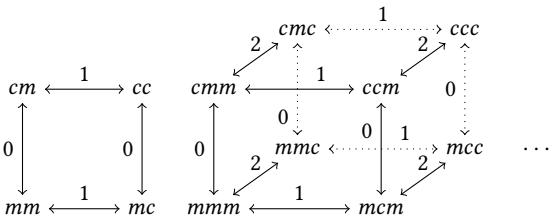

Figure 1: First members of the parameterized Kripke family of the Muddy children example, with parameter 2 (left) and 3 (right), self loops are omitted.

3.1 Parameterized Public Announcement Logic

The logic PPAL will be evaluated on a parameterized Kripke structure. Loosely, such a structure represents a parameterized system, which can be viewed as a union of an infinite family of structures, each obtained by instantiating the parameter. Each state will be assigned a fixed parameter instantation, shared by all its successors. For simplicity, we use only one parameter called the *state size*, which quantifies the (maximal) number of agents involved, as well as the number of copies of atomic propositions.

Definition 3.1. A *parameterized Kripke structure* is a tuple $\mathcal{M} = (S, AP, \rightsquigarrow, L, |\cdot|)$ where:

- S is a (possibly infinite) set of states;
- AP is a finite set of atomic propositions;
- $|\cdot|$ maps any state $s \in S$ to its size $|s| \in \mathbb{N}$;
- L maps any state $s \in S$ and index $i \in [|s|]$ to its *labelling* $L_i(s) \subseteq AP$;
- $\rightsquigarrow \subseteq S \times \mathbb{N} \times S$ is a $\mathbb{N}$-labelled accessibility relation between states, called *indistinguishability relation*, such that any triple $(s, i, s') \in \rightsquigarrow$ satisfies $0 \leq i < |s| = |s'|$. We assume: for any $s \in S$ and $0 \leq i < |s|$, we have $(s, i, s) \in \rightsquigarrow$.

$(s, i, s') \in \rightsquigarrow$ is written $s \overset{i}{\rightsquigarrow} s'$ and reads "if s is the actual state of the system (world), the i-th agent entertains the possibility that the current state is actually s', given its observation." Even though this is not enforced by our definition, most of the proposed models below will assume $\overset{i}{\rightsquigarrow}$ to be an equivalence relation, for all i, and this property will be preserved when deriving models.

Example 3.2. Figure 1 depicts a parameterized Kripke structure for the muddy children puzzle, where $S = \{m, c\}^*$, $AP = \{m\}$, and the size $|w|$ of a state $w \in S$ is defined as its length. For all $i \in [|w|]$, $L_i(w) = \begin{cases} \{m\} \text{ if } w[i] = m \\ \emptyset \text{ otherwise} \end{cases} \in 2^{AP}$

Definition 3.3. We define a formula φ in *parameterized public announcement logic* (PPAL) by the following grammar:

$$\varphi ::= \top \mid \varphi \wedge \varphi \mid \neg\varphi \mid \exists i : \varphi \mid i = 0 \mid i\%k = 0 \mid i = j{+}k \mid p_i \mid \langle i \rangle \varphi \mid \{\varphi!\}\, \varphi$$

Where i, j are index variables, $k \in \mathbb{N}$ is any integral constant and $p \in AP$ is any atomic proposition.

Intuitively, PPAL extends PAL by an indexing capability, so that one could easily refer to the ith agent in the system. This is to some extent akin to how indexed LTL extends LTL [9]. However, we also suitably restrict the indexing capability (essentially, the difference between the indices of two agents is a certain constant k, or that the index of agent is $k \pmod d$ for some constants k and d). This is essentially the extension of the difference logic [20] with modulo operators. This restriction makes the logic amenable to regular model checking techniques, but is also sufficiently powerful for modelling typical examples in parameterized systems.

Shorthands: Boolean connectives $\vee, \rightarrow, \leftrightarrow$ and universal quantification $\forall$ can be encoded in a standard way. The formula $[i]\varphi \equiv \neg\langle i \rangle\neg\varphi$ encodes that agent i knows with certainty that φ holds. Usage of constants is also allowed: $i = k \equiv \exists j : j = 0 \wedge i = j + k$, $p_k \equiv \exists i : i = k \wedge p_i$, $\langle k \rangle\varphi \equiv \exists i : i = k \wedge \langle i \rangle\varphi$.

We denote $FV(\varphi)$ for the set of ("not quantified") *free variables*, of φ. We say that φ is a closed formula whenever $FV(\varphi) = \emptyset$. For any set X of index variables, a function $\mu \in \mathbb{N}^X$ is called a valuation. For a valuation μ and a formula φ, we write $\varphi(\mu)$ for the instantiated formula where each occurrence of $x \in X$ has been replaced by $\mu(x)$. In particular, if $FV(\varphi) \subseteq X$, then $\varphi(\mu)$ is a closed formula.

Definition 3.4. For a parameterized Kripke structure $\mathcal{M}$, a state $s \in S$, a PPAL formula φ, and a valuation $\mu \in \mathbb{N}^{FV(\varphi)}$, we define the *satisfaction relation* $\vDash$, inductively, by $\mathcal{M}, s, \mu \vDash \varphi$ if, and only if, $\forall i, \mu(i) \in [|s|]$ and one of the following condition holds:

$\varphi \equiv \top$

$\varphi \equiv \psi_1 \wedge \psi$ and $\mathcal{M}, s, \mu \vDash \psi_1$ and $\mathcal{M}, s, \mu \vDash \psi_2$

$\varphi \equiv \neg\psi$ and $\mathcal{M}, s, \mu \nvDash \psi$

$\varphi \equiv \exists i : \psi$ and $\mathcal{M}, s, \mu' \vDash \psi$ for some μ' s.t. $\forall x \neq i, \mu(x) \equiv \mu'(x)$

$\varphi \equiv i = j + k$ and $\mu(i) \equiv \mu(j) + k$

$\varphi \equiv i = 0$ and $\mu(i) \equiv 0$

$\varphi \equiv i\%k = 0$ and $\mu(i)\%k \equiv 0$

$\varphi \equiv p_i$ and $p \in L_{\mu(i)}(s)$

$\varphi \equiv \langle i \rangle\psi$ and there exists $t \in S$ such that $s \overset{\mu(i)}{\rightsquigarrow} t$ and $\mathcal{M}, t, \mu \vDash \psi$

$\varphi \equiv \{\psi_1!\}\, \psi_2$ and $\mathcal{M}, s, \mu \vDash \psi_1$ implies $\mathcal{M}\{\varphi(\mu)!\}_\mu, s, \mu \vDash \psi$

where for any closed PPAL formula ψ, $\mathcal{M}\{\psi!\}$ is the (parameterized) Kripke structure $\mathcal{M}$ restricted to the state space satisfying ψ: $S\{\psi!\} = \{s \mid \mathcal{M}, s, \cdot \vDash \psi\}$.

Note that we adopt here the *vacuous truth* semantics for the public announcement operator: whenever a state doesn't satisfy a publicly announced property, it satisfies its conclusion. This choice will turn out to be more convenient with our examples involving the newly iterated public announcement. While an alternative definition $\varphi \wedge \{\varphi!\}\, \psi$ is possible, they are both expressively equivalent.

It is important to notice that the logic does not make a distinction between variables designed for atomic propositions manipulation and variables for indexing agents. Not only this simplification makes our definition more concise, it also enables the specification of relationships between agents and *their* atomic propositions.

Example 3.5. Consider the scenario of the muddy children puzzle, where the father announces that there is *exactly* one muddy child. "after this announcement, every child knows their own state" is encoded as the formula:

$$\left\{\exists i : m_i \wedge \forall j, i \neq j \rightarrow \neg m_j!\right\} \forall i, [i]m_i \vee [i]\neg m_i$$

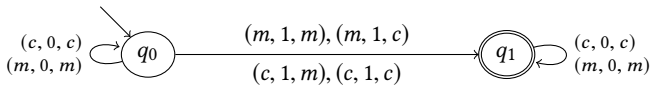

Figure 2: Transducer for the Muddy children

3.2 Regular Kripke Structures

We now provide a regular presentation of parameterized Kripke structures, and define the model checking problem.

Definition 3.6. Let $\mathcal{M} = (S, AP, \rightsquigarrow, L, |\cdot|)$ be a parameterized Kripke structure. It is *regular* if there exists an alphabet Σ such that:

- $S \subseteq \Sigma^*$;
- For all $s \in S$, $|s|$ is the actual length of s, seen as a word;
- For all $i \in [|s|]$, $L_i(s) = L_0(s[i])$;
- The indistinguishability relation can be encoded as a transducer, more precisely the following language is regular:

$$T_{\mathcal{M}} = \left\{ s \otimes V(i, |s|) \otimes t \;\middle|\; s \overset{i}{\rightsquigarrow} t \right\}$$

Recall that we assume the reflexivity $\overset{i}{\rightsquigarrow}$, for each $i \in \mathbb{N}$. Hence, the state space S of a regular Kripke structure is also regular, since $S = \pi_{\Sigma,-}(T_{\mathcal{M}})$ is a morphism image. In the rest of the paper, we will assume the labelling L to be fixed, and identify any regular Kripke structure $\mathcal{M}$ with its regular language $T_{\mathcal{M}}$, seen as a transducer. The following proposition justifies the validity of the above restriction.

Proposition 3.7. *Given an indistinguishability relation $(\overset{i}{\rightsquigarrow})_i$, encoded as a transducer, checking any of the following properties to be satisfied by $\overset{i}{\rightsquigarrow}$ (for each $i \in \mathbb{N}$) is decidable: (1) reflexive, (2) symmetric, and (3) transitive.*

This follows from the fact that reflexivity, symmetry, and transitivity of a binary relation are first-order decidable, and that first-order model checking over regular Kripke structures (more generally *automatic structures*) is decidable [10, 11]. As a remark, it follows also that checking whether a regular Kripke Structure satisfies the S5 axioms (whether all $\overset{i}{\rightsquigarrow}$ are equivalence relations) is decidable.

Example 3.8 (Muddy children). The parameterized Kripke structure of Example 3.2 is regular: the transducer $T_{\mathcal{M}}$ is recognized by the NFA depicted in Figure 2. For example, the accepting run $q_0 \xrightarrow{(c,0,c)} q_0 \xrightarrow{(c,0,c)} q_0 \xrightarrow{(m,1,c)} q_f$ for the word $ccm \otimes 001 \otimes ccc$ encodes the observation $ccm \overset{2}{\rightsquigarrow} ccc$.

The *regular model checking problem for* PPAL is the problem of model checking PPAL formulas over regular Kripke structures: given a regular Kripke structure $\mathcal{M}$, and a formula φ, check if the following satisfaction set is empty:

$$[\![\varphi]\!](\mathcal{M}) := \left\{ (s, \mu) \in S \times \mathbb{N}^{FV(\varphi)} \;\middle|\; \mathcal{M}, s, \mu \vDash \varphi \right\}$$

4 REGULAR MODEL CHECKING OF PPAL

In this section, we prove that this problem is decidable:

Theorem 4.1. *Given a regular Kripke structure $\mathcal{M}$ and a closed* PPAL *formula φ, its semantics $[\![\varphi]\!](\mathcal{M})$ is regular and computable.*

When evaluating a public announcement, the Kripke structure may be modified in a way that is dependent of the current valuation. The crux of the proof lies in carrying a family of regular Kripke structures, encoded as a single extended transducer:

Lemma 4.2. *Let $\mathcal{X}$ be a finite set of variables, φ a* PPAL *formula with $FV(\varphi) \subseteq \mathcal{X}$, and $T \in \mathrm{Reg}\left(\Sigma \times \mathbb{B} \times \Sigma \times \mathbb{B}^{\mathcal{X}}\right)$. We assume that for any $v \in \mathbb{B}^{\mathcal{X}}$, the transducer $\{w \mid w \otimes v \in T\}$ represents a regular Kripke structure denoted $\mathcal{M}_v$. Then, the extended semantics*

$$\widetilde{[\![\varphi]\!]}(T) = \{ s \otimes v \mid \exists \mu \in \mathbb{N}^{\mathcal{X}} : v = V(\mu, |s|) \wedge \mathcal{M}_v, s, \mu \vDash \varphi \}$$

can be recursively computed using boolean, synchronous product and morphism operations on regular languages.

Proof. We focus our proof on two PPAL constructions:

- $\widetilde{[\![\langle a_i \rangle \varphi]\!]}(T) = \pi\left(T \cap A^*BA^* \cap (\Sigma \times \mathbb{B})^* \otimes \widetilde{[\![\varphi]\!]}(T)\right)$

  where: $\begin{cases} A = \Sigma \times \{0\} \times \Sigma \times \{v \mid v(i) = 0\} \\ B = \Sigma \times \{1\} \times \Sigma \times \{v \mid v(i) = 1\} \\ \pi(\alpha, \beta, \gamma, \eta) = (\alpha, \eta) \end{cases}$

  Intuitively, we intersect the transducer with legal moves where the current observational player matches the variable i. We also intersect with the transducer that always ends up in a state and valuation satisfying φ.
- The implementation of the public announcement is by far the most complex one as, we need first to introduce the public announcement transducer $T\{\varphi!\}$, encoding for any v, the regular Kripke structure obtained from $\mathcal{M}_v$, after announcing $\varphi(\mu_v)$:

$$T\{\varphi!\} = \bigcup_{v \in \mathbb{B}^{\mathcal{X}}} \left(T_{\mathcal{M}_v\{\varphi(\mu_v)!\}} \right) \otimes \{v\}$$

  $T\{\varphi!\}$ is actually regular: we first build $\widetilde{[\![\varphi]\!]}(T)$ in order to construct a regular Kripke on this state space. In order to do so, we define the morphism F defined for any[1] $t = w \otimes v \otimes x \otimes w' \otimes v \in (\Sigma \times \mathbb{B}^{\mathcal{X}} \times \Sigma \times \mathbb{B}^{\mathcal{X}})^*$ by $F(t) = w \otimes x \otimes w' \otimes v$. Then, it remains to intersect the image transducer with the initial model: $T\{\varphi!\} = T \cap F(\widetilde{[\![\varphi]\!]}(T) \otimes 0^*10^* \otimes \widetilde{[\![\varphi]\!]}(T))$. Finally, we conclude with the implementation of the (vacuous truth) semantics of the public announcement:

$$\widetilde{[\![\{\varphi!\}\,\psi]\!]} = \widetilde{[\![\neg\varphi]\!]}(T) \cup \widetilde{[\![\psi]\!]}(T\{\varphi!\})$$

□

Example 4.3. Consider again the regular Kripke structure of Figure 2 and the effect of publicly announcing "there is at least one muddy child": initially $\mathcal{M}$ has state space $\Sigma^* = \{m, c\}^*$. After $\{\exists i : m_i!\}$, it is reduced to $\Sigma^*\{m\}\Sigma^*$. After announcing "no one knows (s)he muddy", namely $\{\forall i, \langle i \rangle \neg m_i!\}$, it is further reduced to $\Sigma^*\{m\}\Sigma^*\{m\}\Sigma^*$. And after k similar announcements, the resulting

[1] Notice that the same valuation v appears on both sides.

state space becomes $\Sigma^* (\{m\}\Sigma^*)^k$. This sequence of announcements, however, cannot continue forever as the k-th iteration removes all states of length $k-1$.

As the reader easily infers, the PPAL logic is suitable for the model checking of regular Kripke structures of a given size, but cannot keep up in the parameterized setting, when the number of announcements in the specification depends on the parameter.

Informally, we would like to embed an arbitrary but finite number of iterations, namely an *iterated public announcement* operator [27]:

$$\underbrace{\{\varphi!\}\,\{\varphi!\}\ldots\{\varphi!\}}_{\text{abritrarily many times}}\ \psi$$

Definition 4.4. A formula φ is in PPAL* if it is in the grammar of PPAL, augmentedwith $\varphi ::= \{\varphi!\}^k\,\varphi \mid \{\varphi!\}^*\,\varphi$ with $k \in \mathbb{N}$. The semantics is given by induction on k:

- $[\![\{\varphi!\}^0\,\psi]\!](\mathcal{M}) = [\![\psi]\!](\mathcal{M})$;
- $[\![\{\varphi!\}^{k+1}\,\psi]\!](\mathcal{M}) = [\![\{\varphi!\}^k\,(\{\varphi!\}\,\psi)]\!](\mathcal{M})$;
- $[\![\{\varphi!\}^*\,\psi]\!](\mathcal{M}) = \bigcup_{k\geq 0} [\![\{\varphi!\}^k\,\psi]\!](\mathcal{M})$.

Theorem 4.1 ensures that model checking of a regular Kripke structure against a PPAL formula is decidable, by reduction to regular language universality problem. However, the translation of a formula into a regular language may involve several exponential blow-ups, so the overall running time may become non-elementary. Moreover, this translation does not apply to the newly introduced $\{\cdot!\}^*$ operator, and decidability is not guaranteed in this case. We clarify now these complexity questions:

Theorem 4.5. *There exists a regular structure $\mathcal{M}$, such that:*

(1) *Model checking against a* PPAL *formula is non-elementary;*
(2) *Model checking against a* PPAL* *formula is undecidable.*

Sketch. (1) In [36, Proposition 20], the author constructs an automatic structure $\mathcal{M}$ whose modal logic theory is non-elementary. Modal logic can be seen as a particular fragment of PPAL, with only one agent. An automatic structure can also be seen as a regular Kripke structure with only one agent. The hardness reduction is therefore immediate.

(2) We construct now a regular Kripke structure $\mathcal{M}$ such that its PPAL* theory is undecidable. To this end, we draft a reduction of the Minsky machine halting problem [28]. A configuration of a Minsky machine can be seen as a triple $(q, x_1, x_2) \in \mathbb{N}$, that we encode as the state space of $\mathcal{M}$, relating all states (complete graph). Informally, for any machine C, we define $\varphi_C = \varphi_i \wedge \{\varphi_t!\}^*\,\bot$, which iteratively removes configurations reaching a final state. φ_i encodes the initial configuration while φ_t the existence of a valid transition to another non-final configuration, based on C's definition. The satisfaction set is non-empty if, and only if, the initial configuration is eventually removed (finite run).

□

5 DISAPPEARANCE RELATION

We study in this section the limit behaviour induced by an operator restricting incrementally the state space. This study is motivated by the PPAL* construction $\{\varphi!\}^*\,\bot$, whose semantics can be seen as the set of states *not* being removed, after arbitrarily many state space restrictions operated by the public announcement $\{\varphi!\}$.

For the rest of this section, we fix a more general setting with:

- An initial state space $\mathbb{S} = \mathbb{S}_0 \subseteq \Sigma^*$;
- A function $F : 2^{\Sigma^*} \to 2^{\Sigma^*}$ restricting the state space, that is to say: $\forall X \subseteq \Sigma^*, F(X) \subseteq X$.
- For all $k \in \mathbb{N}$, we let $\mathbb{S}_{k+1} = F(\mathbb{S}_k)$ and $\mathbb{S}_\infty = \cap_{k\geq 0}\mathbb{S}_k$.

Moreover, we assume $\mathbb{S}$ to be a regular language, and F to preserve regular languages, in a way that will later be clarified.

Our study aims at computing the limit set of states $\mathbb{S}_\infty$. Despite our assumptions, this set is not in general regular nor computable, as one can observe as a consequence of the undecidability result of Theorem 4.5, or the following counterexample.

Example 5.1. Consider $\mathbb{S}_0 = \{a\}^*\{b\}^*$, and for all $X \subseteq \mathbb{S}_0$, define $F(X) = \{a^n b^m \mid n = m = 0 \vee (nm > 0 \wedge a^{n-1}b^{m-1} \in X)\}$. Then, for any $k \in \mathbb{N}$, $\mathbb{S}_k = \{a^i b^i \mid 0 \leq i < k\} \cup \{a^{k+n}b^{k+m} \mid n, m \geq 0\}$ is regular, but not its limit $\mathbb{S}_\infty = \{a^i b^i \mid 0 \leq i\}$.

5.1 Unique Characterization

Let us first remark that the application F is not necessarily monotone. Consider for example the announcement $\forall i, m_i \vee [i]\neg m_i$ which reads:

" every non-muddy child knows he's not muddy. "

Then $F(\{cm\}) = \{cm\}$ but $F(\{cm, mm\}) = \emptyset$. As a consequence, $\mathbb{S}_\infty$ is a fixed point of F, but cannot be characterized as the smallest nor the greatest one. Hence, we narrow down our computation goal by introducing the following pre-order over states:

Definition 5.2. The *disappearance* relation $\preceq$ is defined for every $(s,t) \in \mathbb{S}^2$, by:

$$s \preceq t \quad \text{if, and only if,} \quad \forall k \in \mathbb{N}, s \in \mathbb{S}_k \Rightarrow t \in \mathbb{S}_k$$

Intuitively, $s \preceq t$ means that s disappears from the state space before t. $\preceq$ is a total pre-order, i.e. any two elements are comparable, the relation is reflexive, transitive, but not necessarily antisymmetric. Notice that $\mathbb{S}_\infty$ can be characterized as the set of maximal elements of $\preceq$.

In order to reason over set of states induced by a pre-order, we introduce the following notations:

Definition 5.3. For a relation $R \subseteq \mathbb{S} \times \mathbb{S}$, and any $s \in \mathbb{S}$, we define the *upward-closure* and *equivalence class* of s by $\uparrow_R s = \{u \in \mathbb{S} \mid (s,u) \in R\}$, and $[s]_R = \{u \in \mathbb{S} \mid (s,u) \in R \wedge (u,s) \in R\}$, respectively. We omit the subscript notation when $R = \preceq$.

As suggested by its name, the latter notion involves an equivalence relation, namely $R \cap R^{-1}$ which relates states of $\mathbb{S}$ disappearing at the same iteration. On the other hand, the upward-closure $\uparrow s$ can be interpreted as one of the iterated $\mathbb{S}_k = \uparrow s$ for some $k \in \mathbb{N} \uplus \{\infty\}$. When $k < \infty$, we know this is the *last* iteration before s and all its equivalent states got removed. This entails $\mathbb{S}_{k+1} = \mathbb{S}_k \backslash [s]$, hence $F(\uparrow s) = \uparrow s \backslash [s]$. When $k = \infty$, we know on the contrary, that s never disappears, which also means $[s] = \uparrow s = \mathbb{S}_\infty$.

This setting is summarised in the following Figure 3 and Proposition 5.4, the latter also provides a unique characterization under certain conditions:

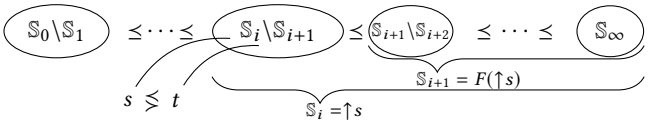

Figure 3: Hierarchy of the equivalence classes of the disappearance relation

Proposition 5.4. *Let $R \subseteq \mathbb{S} \times \mathbb{S}$. If $R = \preceq$, then:*

(1) *R is a total pre-order on $\mathbb{S}$, and*
(2) *for $s \in \mathbb{S}$, $[s]_R = \uparrow_R s \backslash F(\uparrow_R s)$ or $[s]_R = \uparrow_R s = F(\uparrow_R s)$.*

Moreover, the converse holds whenever $\mathbb{S}$ is finite.

5.2 Learning Procedure

The unique characterization of Proposition 5.4 paves the way to a learning procedure for computing $\preceq$. More precisely, we consider for this section an encoding of $\preceq$ seen as as a language over pairs of letters: $L_\preceq = \{s \otimes t \mid |s| = |t| \wedge s \preceq t\} \subseteq (\Sigma \times \Sigma)^*$
Assuming $L_\preceq$ is a regular language, we will develop a learning procedure to construct it. On the one hand, notice that this definition of $L_\preceq$ looses some information about $\preceq$ as it can only relate states of the same length. On the other hand, this restriction is not crucial as the PPAL logic is exclusively based on length-preserving transducers. We keep the following requirement:

($R1$): "F is length-preserving"

Strictly speaking, we assume $L_\preceq$ to be the representation of the family $(\preceq_k)_{k\in\mathbb{N}}$, where for each k, $\preceq_k$ is the disappearance relation starting from the initial state space $\mathbb{S}_0 \cap \Sigma^k$. As Σ^k is finite for a given k, this restriction further allows us to provide a unique characterization of $L_\preceq$, as provided by Proposition 5.4.

We now introduce the L^* algorithm from Angluin, which allows us to learn a finite automaton $\mathcal{A}$, or equivalently a target regular language $L_t \in \mathrm{Reg}(\Sigma_t)$, based on queries answered by an Oracle. Such an Oracle has to answer *membership* and *equivalence* queries, by direct access to the target language or by indirect means.

We explain the exact semantics of L^* queries for a target language language $L_t \in \mathrm{Reg}(\Sigma_t)$, and how they are answered in this learning procedure, where the target language is $L_\preceq \in \mathrm{Reg}(\Sigma \times \Sigma)$:

- **Membership Queries:** the Oracle is asked whether a given word $w \in \Sigma_t^*$ is in the target language L_t.
  Answer: we let $s, t \in \Sigma^{|w|}$ with $s \otimes t = w$ and decide whether $s \preceq t$. We proceed to the iterative computation of the sets $(\mathbb{S}_k)_{k\in\mathbb{N}}$ and stop whenever s or t is no more in the set. This is however a semi-decision procedure as it may fail in the case where neither s nor t disappear ($s, t \in \mathbb{S}_\infty$). To circumvent this issue, we perform the computation on the restricted state space of a fixed length $|w|$, namely $\mathbb{S}_k \cap \Sigma^{|w|}$, ensuring a finite cardinality. As soon as $s, t \in \mathbb{S}_k \cap \Sigma^{|w|} = \mathbb{S}_{k+1} \cap \Sigma^{|w|}$, we conclude that $s \preceq t$. This leads to our second requirement:

  ($R2$): "F restricts the state space independently for different state sizes."

- **Equivalence Queries:** Given a candidate language L, the Oracle is asked whether $L = L_t$ and if not, provides a counterexample $w \in L\backslash L_t \cup L_t\backslash L$.
  Answer: we use Proposition 5.4, which can be seen as a first order characterization of $\preceq$, and translate the listed criteria into equivalence problems over regular languages. If one regular language equivalence fails, we have to provide a counter example to the learning procedure. Unfortunately, a counterexample to a criterion of Proposition 5.4 does not directly provide a counterexample for L^*. For example, a counterexample for the transitivity property would consist in a triple $(s_1, s_2, s_3) \in \mathbb{S}^3$ such that $s_1 \otimes s_2 \in L$, $s_2 \otimes s_3 \in L$ but $s_1 \otimes s_3 \notin L$, and it wouldn't be clear whether the property fails because either (s_1, s_2) or (s_2, s_3) should be removed from L or because (s_1, s_3) should be added. Nonetheless, since a counterexample was provided for a fixed length l, L restricted to $(\Sigma\times\Sigma)^l$ is guaranteed not to be a proper encoding of $\preceq \cap \Sigma^l \times \Sigma^l$. A direct enumeration of the sequence $(\mathbb{S}_k \cap \Sigma^l)_{k\geq 0}$ will therefore terminate and provide a counterexample.

5.3 Effective and Uniform Regularity

In order to effectively implement the procedure, we provide the following equivalent characterization of Proposition 5.4, in terms of first-order formulae.

Proposition 5.5. *Let $R \subseteq \Sigma^* \times \Sigma^*$ and $k \in \mathbb{N}$.*
$R \cap (\Sigma^k \times \Sigma^k) \neq \preceq_k$ if, and only if, any *one of the conditions holds:*

(1) $\exists s, t : (s, t) \in R \wedge (s, t) \notin \mathbb{S} \times \mathbb{S}$;
(2) $\exists s : (s, s) \notin R$;
(3) $\exists s_1, s_2, s_3 : (s_1, s_2) \in R \wedge (s_2, s_3) \in R \wedge (s_1, s_3) \notin R$;
(4) $\exists s, t : (s, t) \notin R \wedge (t, s) \notin R$;
(5)
$$\exists s, t_1, t_2 : \begin{cases} (s, t_1) \in R \wedge (s, t_2) \in R \\ (t_1, s) \notin R \leftrightarrow t \in F(\uparrow_R s) \\ (t_2, s) \notin R \vee t_2 \notin F(\uparrow_R s) \end{cases}$$

Where all quantifications are made over Σ^k.

Based on this first-order characterization, we provide an actual implementation of equivalence queries on the candidate language L, by resorting to queries on length-preserving transducers, namely regular languages over $\Sigma\times\Sigma$. For example, Property (1) is translated to the query $L \cap \overline{\mathbb{S} \otimes \mathbb{S}} \stackrel{?}{=} \emptyset$.

While the predicates $\mathbb{S} \times \mathbb{S}$ and R can be encoded as the regular languages $\mathbb{S} \otimes \mathbb{S}$ and L_c, respectively, property (5) involves the computation of the operator F as the following binary predicate:

$$F(\uparrow_R \cdot) = \{(s, t) \mid s \in F(\uparrow_R t)\}$$

This condition is introduced as the last requirement:

($R3$): "F is effective and uniformly regular"

Conditions ($R1$) – ($R3$) are formally defined through the following conditions:

Definition 5.6. Let G be a function from $2^{\Sigma_1^*}$ to $2^{\Sigma_2^*}$.

- G is *independently length-preserving* if:
  $\forall l \in \mathbb{N}\ \forall X \subseteq \Sigma_1^*, G(X \cap \Sigma_1^l) = G(X) \cap \Sigma_2^l$;

- G is *effectively uniformly regular* if:
  For any given alphabet Σ' and $L \in \mathrm{Reg}(\Sigma' \times \Sigma_1)$, the following language is regular and computable:

$$\left\{ w' \otimes w_2 \;\middle|\; \exists w_1 \in \Sigma^{|w_2|} : w_2 \in G\left(\{w_1 \mid w' \otimes w_1 \in L\}\right) \right\}$$

Theorem 5.7. *Assume F is an independently length-preserving and uniformly regular function.*
Then the L^* learning procedure described in Section 5.2 eventually terminates and returns $L_\preceq$ if, and only, it is regular.

5.4 Application to PPAL*

We finally address the general case with the following observation:

Proposition 5.8. *Let φ and ψ be two closed formula and $\mathcal{M}$ a parameterized Kripke structure, whose state space is $\mathbb{S}$. For any $X \subseteq \mathbb{S}$, we define $\mathcal{M}_{|X}$ for the parameterized Kripke structure restricted to X and consider the resulting disappearance relation $\preceq \subseteq \mathbb{S}^2$.*
We have:

$$[\![[\varphi!]^*\psi]\!](\mathcal{M}) = \left\{ s \in \mathbb{S} \;\middle|\; \exists t \in \mathbb{S} : s \in [\![\psi]\!](\mathcal{M}_{|\uparrow_\preceq t}) \right\}$$

We can easily see that the above set is regular if $\mathcal{M}$ and $\preceq$ are both regular. In order to proceed to their computation, we need to provide the following uniformly regular property:

Proposition 5.9. *Let φ be a closed formula on a regular Kripke structure $\mathcal{M}$. The application, F_φ defined by*

$$\forall X \subseteq \mathbb{S}, F_\varphi(X) = [\![\varphi]\!](\mathcal{M}_{|X})$$

is length-preserving, effectively and uniformly regular.

Proof. Given $L \in \mathrm{Reg}(\Sigma' \times \Sigma)$, we define a new regular Kripke structure $\mathcal{M}'$ storing the information about Σ' in its state space. The construction of $\mathcal{M}'$ is effective, and by Theorem 4.1, we can compute $[\![\varphi]\!](\mathcal{M}')$. □

6 EXPERIMENTS

We developed a prototype tool implementation, using the Java libraries Learnlib and Automatalib [16]. Three different models were specified then verified[2] showing tractability of the procedure:

Model	Duration	Memory Usage
Russian cards	36s	2365MB
Large number	53s	1218MB
$M \leq 9$ Muddy children	24s	1136Mo
$M \leq 10$ Muddy children	TO (5min+)	
$M < \infty$ Muddy children	2.5s	111MB

The rest of the section discusses implementation details and description of the aforementioned models.

Usage. The tool takes as an input an automaton description of a regular Kripke structure $\mathcal{M}$, and for each specification φ, computes its satisfaction set. In case the complement $[\![\neg\varphi]\!](\mathcal{M})$ is non-empty, a NFA is returned, which can be interpreted as the set of counterexamples to φ. For usability reasons, the syntax of PPAL* is enhanced with several syntactic sugars, but can also embed dummy formulae, equivalent to $\top$, whose evaluation triggers visualization of the intermediate constructed automata.

[2]Experiments were conducted on a i7-8550U CPU @ 1.80GHz, 16GB machine with JavaSE-1.8. The prototype and model files are available online [34].

Automaton size. Since specifying a transducer for $\dot{\rightsquigarrow}$ can be quite tedious, we specify a rather general regular Kripke structure encoding only the observation of the agents, and further restricting the state space by applying public announcement constructions. As a matter of fact, the state space after only few announcements can already require several hundred states. The intermediate computations may even lead to semantics automata of up to millions of states. Note that the ordering of index quantifications inside the specification plays a crucial role, as each quantified index is carried around in one coordinate of the automaton alphabet, as explained by Lemma 4.2.

Learning procedure. Although several DFA learning algorithms are provided by Learnlib, the classical Angluin's L^*turned out to be sufficient for our experiments: for all our examples, whenever termination was guaranteed[3], the algorithm converged within a minute. The most expensive task of the equivalence check is the last property of Proposition 5.5: it is indeed the only criterion involving the evaluation of the PPAL formula. Fortunately, many equivalence queries fail on previous criteria, that are less expensive to check.

6.1 Russian Card Problem

This puzzle [40] involves N different cards which are distributed between three players Alice, Bob and Cathy. The goal of the game is for Alice and Bob to exchange messages publicly, in order to get to know who has which card in their hand, without disclosing any individual card information to Cathy.

In the one-round setting, Alice broadcasts a first message, then Bob replies, which conclude the protocol. As Bob can only announce a piece of information he already knows, his message can trivially be assumed to announce Cathy's cards. In other words, the one-round case focuses on Alice's announcement.

Kripke structure. We let $AP = \{a, b, c\}$, and the only agent indexes involved[4] are $a = 1$, $b = 2$, and $c = 3$. For $x \in AP$ and $i \in \mathbb{N}$, x_i holds iff agent x has card i in their hand. Moreover, we assume that a_i, b_i and c_i are mutually exclusive (each card appears only in one) hand. We easily check that $\mathcal{M}$ is regular.

Specification. An announcement of Alice is any statement about her own observation, namely a characterization of the cards in her hand, or equivalently $\{\{i_0, i_1, i_2\}, \{i_3, i_4, i_5\}, \ldots\}$, seen as a set of possible hands. However, this representation is not fit to a parameterized context, where the total number of cards is not fixed a priori. Instead, we consider announcements specified in a parameterized manner, namely in the propositional fragment of PPAL, involving only index quantifications, the atomic proposition a and no epistemic operator. A formula ψ is a *good announcement* if furthermore, it satisfies:

$$\begin{aligned}
\varphi_{good} = \quad & \psi \wedge \{\psi!\}(&& \text{// truthful PA} \\
& \forall i, [b]a_i \vee [b]b_i \vee [b]c_i && \text{// b knows the distribution} \\
& \forall i, \neg c_i \rightarrow \begin{cases} \langle c\rangle a_i \wedge \langle c\rangle \neg a_i \\ \langle c\rangle b_i \wedge \langle c\rangle \neg b_i \end{cases}) && \text{// c doesn't know}
\end{aligned}$$

While [4] provides several sufficient and necessary conditions on the number of cards received by each participants, we focus here on a single example of (sufficient) good announcement, provided

[3]The learning procedure diverges if, and only if, $L_\preceq$ is not regular.

[4]We can assume that $\overset{i}{\rightsquigarrow}$ is trivial for $i \geq 3$.

by [4, Proposition 5] in the case where $N\%3 = 0$, Alice receives 3 cards, Cathy only one, and Bob the rest (property φ_{model}).

If Alice received the first three cards, the following announcement is claimed to be good:

$$\psi \equiv \exists j : j\%3 = 0 \wedge \big((a_j \wedge a_{j+1} \wedge a_{j+2}) \vee (a_j \wedge a_{j+4} \wedge a_{j+8})\big)$$

Which can be checked with the verification question:

$$\mathcal{M} \overset{?}{\models} \{\varphi_{model}!\}\left(a_0 \wedge a_1 \wedge a_2 \rightarrow \varphi_{good}\right)$$

We leave to the reader the generalization to any initial hand of Alice and the specification of φ_{model}.

6.2 Highest Number

The highest number problem involves two agents Alice and Bob both receiving a different natural number between 0 and N, which they keep private. We model this situation by $AP = \{a, b\}$ and encode the observation of $a = 0$ and $b = 1$ as a transducer. A letter $\alpha \in AP$ is used to encode α's number, in unary. At each round, they are both asked simultaneously if one of them knows who has the highest number. If not, a public announcement is made for this fact. If yes, the game stops. The termination of this protocol is checked by the following iterated announcement, which we successfully verify: $\{\neg(\exists i \exists j : [j](a_i \wedge \neg b_i) \vee [j](\neg a_i \wedge b_i))!\}^* \bot$

6.3 Muddy Children: Bounded Case

In this section, assume the number of muddy children is bounded by some fixed $M \in \mathbb{N}$, although the total number of children is left as a parameter N. This assumption is implemented as public annonuecement made on the regular Kripke structure of Example 3.8.

Intuitively, the effect of this announcements is to construct the product automaton of the original transducer $T_{\mathcal{M}}$ with a finite automaton of size $M + 1$ counting how many muddy children have been seen so far. This product has to be made twice: once on the source and once on the target of the transducer. Nonetheless, the target and source word differ only by one letter, hence the resulting automaton is of size $O(M)$.

Then, we proceed to the iterated announcement $\{\forall i, \langle i\rangle\neg m_i!\}^*$, which reduces to the disappearance relation computation: for $s, t \in \mathbb{S}_0$, $s \preceq t$ if, and only if, $|s|_m \leq |t|_m$. As a matter of fact, the protocol terminates after $|s|_m$ announcements of the father whenever there are exactly $|s|_m$ muddy children.

This relation can be effectively encoded as a length-preserving transducer, counting the difference of number of muddy children between s and t, which lies between $-M$ and M. Our algorithm successfully computes a transducer for $\preceq$, with $O(M)$ states.

6.4 Muddy Children: Unbounded Case

We remove now the boundedness condition. As before, $\preceq$ has to compare the number of muddy children between two given states, which can now be arbitrarily large: take for example $m^n c^n \preceq c^n m^n$. As a consequence, $L_{\preceq}$ is not regular anymore and the learning procedure doesn't terminate.

Nonetheless, we observe that the problem is invariant under permutation, more precisely: (1) The formula φ lies in a fragment of PPAL* without index comparison of the form $i = j + k$ for any $k \neq 0$; and (2) For any word $w \in T_{\mathcal{M}}$ and any bijection Σ on $[|w|]$, $w[\sigma(0)] \dots w[\sigma(|w| - 1)] \in T_{\mathcal{M}}$.

Therefore, we proceed to a counting abstraction of the model, restricting the regular Kripke structure. Informally, we want to preserve the property that a transition $s \overset{i}{\rightsquigarrow} t$ is valid if, and only if, there exists some agent j with the same "local state" as i, that can perform this transition. Here, the announcement translates to "there is still a muddy child who doesn't know".

As the state space is reduced to c^*m^*, our rewriting actually consists in a unary encoding of the number of clean and muddy children. As for the largest number challenge, the disappearance relation is regular, and we successfully verify the rewritten formula:

$$\varphi \equiv \{\forall i, \neg m_{i+1} \rightarrow \neg m_i!\}\,\{\exists i : m_i!\}\,\{\exists i : m_i \wedge \langle i\rangle\neg m_i!\}^* \bot$$

7 RELATED AND FUTURE WORK

Related Work. Finite-state model checkers for various epistemic logics are available, e.g., MCMAS [26], DEMO [39, 44], SMCDEL [38], and MCK [14]. Kouvaros and Lomuscio [19] have studied *cutoff techniques* for ACTL*K\X, a temporal-epistemic logic combining S5 and temporal logic ACTL* \ X, which is used in MCMAS. Roughly speaking, a cutoff exists for a parameterized system when the behavior of any instance of the system can be simulated (using an appropriate notion of simulation) by the behavior of systems of a fixed computable parameter-size k, which would allow us to reduce the parameterized model checking problem into finite-state model checking (up to parameter of size k). This cutoff technique — as is the case with most cutoff methods (see [9, 45]) — needs to be specially tuned to different subclasses of parameterized systems. We are not aware of the existence of such cutoff values for the systems that we consider in this paper. Our regular model checking method is complementary to such cutoff methods. The method is fully automatic, but it might not terminate in general (albeit we provide also termination guarantees). To the best of our knowledge, our method provides the first automatic solution to the parameterized verification problem for the muddy children puzzle, the Russian card puzzle [40], and the large number challenge, all of which have been studied in the finite-state case (e.g. see [26, 38, 39, 44]).

Future Work. Natural extensions of PPAL* include the support of dynamic properties, enabling the specification and verification of richer communication protocols, where the communication pattern is non-deterministic [26]. The study of the disappearance relation revealed that the chosen encoding is crucial for termination. The counting abstraction sketched for the Muddy children case would benefit from a systemic approach. Once the symmetries have been detected in the automatic structure, which can be implemented [22] with transducer techniques, a lossless Parikh image [30] could be computed in terms of a Presburger formula [32]. As the PPAL semantics involves only boolean, synchronous product and morphism operations, the computation could be performed in this domain. As another future direction, we would like to study cutoff methods of [19] for the examples that we consider in this paper.

ACKNOWLEDGMENTS

This work was supported by the ERC Starting Grant 759969 (AVSMP) and Max-Planck Fellowship.

REFERENCES

[1] Parosh Aziz Abdulla. 2012. Regular model checking. *Int. J. Softw. Tools Technol. Transf.* 14, 2 (2012), 109–118. https://doi.org/10.1007/s10009-011-0216-8

[2] Parosh Aziz Abdulla, Frédéric Haziza, and Lukás Holík. 2016. Parameterized verification through view abstraction. *STTT* 18, 5 (2016), 495–516.

[3] Parosh Aziz Abdulla, Bengt Jonsson, Marcus Nilsson, and Mayank Saksena. 2004. A Survey of Regular Model Checking. In *CONCUR 2004 - Concurrency Theory, 15th International Conference, London, UK, August 31 - September 3, 2004, Proceedings (Lecture Notes in Computer Science)*, Philippa Gardner and Nobuko Yoshida (Eds.), Vol. 3170. Springer, 35–48. https://doi.org/10.1007/978-3-540-28644-8_3

[4] Michael H. Albert, Robert E. L. Aldred, Mike D. Atkinson, Hans P. van Ditmarsch, and Chris C. Handley. 2005. Safe communication for card players by combinatorial designs for two-step protocols. *Australas. J Comb.* 33 (2005), 33–46. http://ajc.maths.uq.edu.au/pdf/33/ajc_v33_p033.pdf

[5] Benjamin Aminof, Aniello Murano, Sasha Rubin, and Florian Zuleger. 2016. Automatic Verification of Multi-Agent Systems in Parameterised Grid-Environments. In *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems, Singapore, May 9-13, 2016*, Catholijn M. Jonker, Stacy Marsella, John Thangarajah, and Karl Tuyls (Eds.). ACM, 1190–1199. http://dl.acm.org/citation.cfm?id=2937098

[6] Dana Angluin. 1987. Learning Regular Sets from Queries and Counterexamples. *Inf. Comput.* 75, 2 (Nov. 1987), 87–106. https://doi.org/10.1016/0890-5401(87)90052-6

[7] Krzysztof R. Apt and Dexter Kozen. 1986. Limits for Automatic Verification of Finite-State Concurrent Systems. *Inf. Process. Lett.* 22, 6 (1986), 307–309. https://doi.org/10.1016/0020-0190(86)90071-2

[8] Francesco Belardinelli and Alessio Lomuscio. 2009. Quantified epistemic logics for reasoning about knowledge in multi-agent systems. *Artif. Intell.* 173, 9-10 (2009), 982–1013. https://doi.org/10.1016/j.artint.2009.02.003

[9] Roderick Bloem, Swen Jacobs, Ayrat Khalimov, Igor Konnov, Sasha Rubin, Helmut Veith, and Josef Widder. 2015. *Decidability of Parameterized Verification.* Morgan & Claypool Publishers. https://doi.org/10.2200/S00658ED1V01Y201508DCT013

[10] A. Blumensath. 1999. *Automatic Structures.* Master's thesis. RWTH Aachen.

[11] A. Blumensath and E. Grädel. 2004. Finite Presentations of Infinite Structures: Automata and Interpretations. *Theory Comput. Syst.* 37, 6 (2004), 641–674.

[12] Yu-Fang Chen, Chih-Duo Hong, Anthony W. Lin, and Philipp Rümmer. 2017. Learning to prove safety over parameterised concurrent systems. In *2017 Formal Methods in Computer Aided Design, FMCAD 2017, Vienna, Austria, October 2-6, 2017*. 76–83.

[13] Ronald Fagin, Joseph Y. Halpern, Yoram Moses, and Moshe Vardi. 1995. *Reasoning about Knowledge.* MIT Press.

[14] Peter Gammie and Ron van der Meyden. 2004. MCK: Model Checking the Logic of Knowledge. In *Computer Aided Verification, 16th International Conference, CAV 2004, Boston, MA, USA, July 13-17, 2004, Proceedings*. 479–483. https://doi.org/10.1007/978-3-540-27813-9_41

[15] Nina Gierasimczuk and Jakub Szymanik. 2011. A Note on a Generalization of the Muddy Children Puzzle. In *Proceedings of the 13th Conference on Theoretical Aspects of Rationality and Knowledge (TARK XIII)*. Association for Computing Machinery, New York, NY, USA, 257–264. https://doi.org/10.1145/2000378.2000409

[16] Malte Isberner, Falk Howar, and Bernhard Steffen. 2015. The Open-Source LearnLib. In *Computer Aided Verification*, Daniel Kroening and Corina S. Păsăreanu (Eds.). Springer International Publishing, Cham, 487–495.

[17] Michael J. Kearns and Umesh V. Vazirani. 1994. *An Introduction to Computational Learning Theory.* MIT Press. https://mitpress.mit.edu/books/introduction-computational-learning-theory

[18] Panagiotis Kouvaros and Alessio Lomuscio. 2013. Automatic verification of parameterised multi-agent systems. In *International conference on Autonomous Agents and Multi-Agent Systems, AAMAS '13, Saint Paul, MN, USA, May 6-10, 2013*, Maria L. Gini, Onn Shehory, Takayuki Ito, and Catholijn M. Jonker (Eds.). IFAAMAS, 861–868. http://dl.acm.org/citation.cfm?id=2485057

[19] Panagiotis Kouvaros and Alessio Lomuscio. 2016. Parameterised verification for multi-agent systems. *Artificial Intelligence* 234 (2016), 152 – 189. https://doi.org/10.1016/j.artint.2016.01.008

[20] Daniel Kroening and Ofer Strichman. 2008. *Decision Procedures.* Springer.

[21] Ondrej Lengál, Anthony Widjaja Lin, Rupak Majumdar, and Philipp Rümmer. 2017. Fair Termination for Parameterized Probabilistic Concurrent Systems. In *Tools and Algorithms for the Construction and Analysis of Systems - 23rd International Conference, TACAS 2017, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2017, Uppsala, Sweden, April 22-29, 2017, Proceedings, Part I*. 499–517. https://doi.org/10.1007/978-3-662-54577-5_29

[22] Anthony W. Lin, Truong Khanh Nguyen, Philipp Rümmer, and Jun Sun. 2016. Regular Symmetry Patterns. In *Verification, Model Checking, and Abstract Interpretation*, Barbara Jobstmann and K. Rustan M. Leino (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 455–475.

[23] Anthony W. Lin and Philipp Rümmer. 2016. Liveness of Randomised Parameterised Systems under Arbitrary Schedulers. In *Computer Aided Verification - 28th International Conference, CAV 2016, Toronto, ON, Canada, July 17-23, 2016, Proceedings, Part II*. 112–133.

[24] Anthony W. Lin and Philipp Rümmer. 2021. Regular Model Checking Revisited. *To appear in Bengt Jonsson's 60 Festschrift* (2021). https://arxiv.org/abs/2005.00990

[25] Alessio Lomuscio and Edoardo Pirovano. 2020. Parameterised Verification of Strategic Properties in Probabilistic Multi-Agent Systems. In *Proceedings of the 19th International Conference on Autonomous Agents and Multiagent Systems, AAMAS '20, Auckland, New Zealand, May 9-13, 2020*, Amal El Fallah Seghrouchni, Gita Sukthankar, Bo An, and Neil Yorke-Smith (Eds.). International Foundation for Autonomous Agents and Multiagent Systems, 762–770. https://dl.acm.org/doi/abs/10.5555/3398761.3398852

[26] Alessio Lomuscio, Hongyang Qu, and Franco Raimondi. 2017. MCMAS: An Open-Source Model Checker for the Verification of Multi-Agent Systems. *Int. J. Softw. Tools Technol. Transf.* 19, 1 (Feb. 2017), 9–30. https://doi.org/10.1007/s10009-015-0378-x

[27] Joseph S. Miller and Lawrence S. Moss. 2005. The Undecidability of Iterated Modal Relativization. *Stud Logica* 79, 3 (2005), 373–407. https://doi.org/10.1007/s11225-005-3612-9

[28] Marvin Minsky. 1967. *Computation: Finite and Infinite Machines.* Prentice Hall International.

[29] Daniel Neider and Nils Jansen. 2013. Regular Model Checking Using Solver Technologies and Automata Learning *(Lecture Notes in Computer Science)*, Guillaume Brat, Neha Rungta, and Arnaud Venet (Eds.).

[30] Rohit J. Parikh. 1966. On Context-Free Languages. *J. ACM* 13, 4 (Oct. 1966), 570–581. https://doi.org/10.1145/321356.321364

[31] Jan Plaza. 2007. Logics of public communications. *Synth.* 158, 2 (2007), 165–179. https://doi.org/10.1007/s11229-007-9168-7

[32] Helmut Seidl, Thomas Schwentick, Anca Muscholl, and Peter Habermehl. 2004. Counting in Trees for Free. In *Automata, Languages and Programming*, Josep Díaz, Juhani Karhumäki, Arto Lepistö, and Donald Sannella (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 1136–1149.

[33] Michael Sipser. 1997. *Introduction to the Theory of Computation.* PWS Publishing Company.

[34] Daniel Stan and Anthony W. Lin. 2021. MCPPAL: Regular Model Checking for Parametric Public Announcement Logic (Artifact). https://zenodo.org/record/4507467. (2021). https://doi.org/10.5281/zenodo.4507467 Source https://arg-git.informatik.uni-kl.de/ds/mcppal.

[35] Daniel Stan and Anthony Widjaja Lin. 2021. Regular Model Checking Approach to Knowledge Reasoning over Parameterized Systems. (2021). arXiv:cs.FL/2102.04361

[36] Anthony Widjaja To. 2009. Model Checking FO(R) over One-Counter Processes and beyond. In *Computer Science Logic*, Erich Grädel and Reinhard Kahle (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 485–499.

[37] Anthony Widjaja To and Leonid Libkin. 2010. Algorithmic Metatheorems for Decidable LTL Model Checking over Infinite Systems. In *FoSSaCS*. 221–236. https://doi.org/10.1007/978-3-642-12032-9_16

[38] Johan van Benthem, Jan van Eijck, Malvin Gattinger, and Kaile Su. 2018. Symbolic model checking for Dynamic Epistemic Logic - S5 and beyond. *J. Log. Comput.* 28, 2 (2018), 367–402. https://doi.org/10.1093/logcom/exx038

[39] Hans Van Dimarsch and Ji Ruan. 2007. Model Checking Logic Puzzles. (Nov. 2007). https://hal.archives-ouvertes.fr/hal-00188953 working paper or preprint.

[40] Hans P. van Ditmarsch. 2003. The Russian Cards Problem. *Stud Logica* 75, 1 (2003), 31–62. https://doi.org/10.1023/A:1026168632319

[41] Hans P. van Ditmarsch, Ji Ruan, and L. C. Verbrugge. 2005. Model Checking Sum and Product. In *AI 2005: Advances in Artificial Intelligence, 18th Australian Joint Conference on Artificial Intelligence, Sydney, Australia, December 5-9, 2005, Proceedings (Lecture Notes in Computer Science)*, Shichao Zhang and Ray Jarvis (Eds.), Vol. 3809. Springer, 790–795. https://doi.org/10.1007/11589990_82

[42] Hans P. van Ditmarsch, Wiebe van der Hoek, and Barteld Kooi. 2008. *Dynamic Epistemic Logic.* Springer.

[43] Hans P. van Ditmarsch, Wiebe van der Hoek, Ron van der Meyden, and Ji Ruan. 2006. Model Checking Russian Cards. *Electron. Notes Theor. Comput. Sci.* 149, 2 (2006), 105–123. https://doi.org/10.1016/j.entcs.2005.07.029

[44] J. van Eijck. 2014. DEMO-S5. (2014). Tech. rep., CWI.

[45] Tomas Vojnar. 2007. Cut-offs and Automata in Formal Verification of Infinite-State Systems. (2007). Habilitation Thesis, Faculty of Information Technology, Brno University of Technology.

[46] Lenore D. Zuck and Amir Pnueli. 2004. Model checking and abstraction to the aid of parameterized systems (a survey). *Computer Languages, Systems & Structures* 30, 3-4 (2004), 139–169. https://doi.org/10.1016/j.cl.2004.02.006