Regular Model Checking Approach to Knowledge Reasoning 
over Parameterized Systems (technical report)

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

1 INTRODUCTION
Consider the standard problem of muddy children puzzle in knowledge reasoning [11]. 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 the number of muddy children (i.e. a child can observe whether another child (but not himself) has a mud on their forehead). The muddy children protocol goes in rounds. 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 [11]).

The muddy children puzzle, as stated above, can be constructed as a typical example of a parameterized verification problem [4, 8, 16, 17] but with respect to epistemic properties. Even though the problem was shown to be decidable for a simple safety property by Apt and Kozen in the 80s [6], the past twenty years or so have witnessed a lot of progress in the field of parameterized verification (e.g., see [1, 8, 38, 39] for excellent surveys). Researchers resort to either (1) general 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 [4, 8, 16, 17, 20]). Despite this, very little work has been done on parameterized verification problem with respect to epistemic properties, in particular which is applicable in the simple setting of the muddy children example. This is an extremely challenging problem, while most of the research focus on parameterized system verification for a few decades is on simple safety properties, and only recently on liveness properties.

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) [25, 35] (also see [32, 34, 36], 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 \( \sim^a \) (for each agent \( a \)) satisfying the S5 axioms, i.e., \( \sim^a \) is an equivalence relation (reflexive, symmetric, and transitive). That way, \( \sim^a \) 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 \( \langle \varphi \rangle \), whereby each agent learns about \( \varphi \). A standard application of the public 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) [11, 25, 35] 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 do exist (e.g., quantified epistemic logic [7], iterated public announcement [13, 22]; see §5). To the best of our knowledge, 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, 2, 9, 10, 30], wherein a configuration in the (parameterized) systems are represented by a string over some finite alphabet Σ, while a binary relation ∼⊆ Σ × Σ is represented by an automata over the product alphabet Σ × Σ. [The reader could understand a product automaton just like a normal automaton, where an automaton would synchronously read a pair (a, b) of symbols at each step.] The corresponding Kripke structures are called automatic Kripke structures [9, 10, 30]. One benefit of this framework is that one could encode an infinite number of accessibility relation ∼(i) ⊆ Σ × Σ for each agent indexed i = 0, 1, 2, ..., where a single automaton representing ∼⊆ Σ × N × Σ. Since a string encoding s(i) of each number i ∈ N could be given (e.g., i = 1 could be represented in unary), the automaton could run over some product alphabet, e.g., Σ × (0, 1) × Σ. Second, to reason about knowledge over automatic Kripke Structures, it is important to enrich PAL with synchronous product and morphism:

For w1 ∈ Σ1 and w2 ∈ Σ2, we extend f to words over Σ1 by defining, for any w ∈ Σ1, f(w) = f(w[0])...f(w[|w|−1]) ∈ Σ∗, then to languages over the super-set Σ∗, for any L ⊆ Σ∗, f(L) = {f(w) | w ∈ L ∩ (Σ1)∗}. Of particular interest, we define projection morphisms:

For example, synchronous product’s counterparts can be defined as the morphisms πi(Σ1,...,Σn) and πi−1(Σ1,...,Σn), projections on the first and second component, respectively.

We encode positions inside a word with the alphabet B = {0, 1}, for 0 ≤ i < l, V(i, l) = 0i1l−i−1 ∈ Bl encodes the i-th position. When the meaning is clear, we will at times identify a finite automaton A and its recognized language L(A) ∈ Reg(Σ). In particular, whenever we claim a language L is regular, a recognizing automaton may be provided instead. Whenever Σ = Σ1 × Σ2, the automaton may also be called a length-preserving transducer, or simply “transducer”, as it can be interpreted as an automaton mapping a word w1 ∈ Σ1 to (non-deterministically) a word w2 ∈ Σ2 of the same length, such that w1 ⊗ w2 ∈ 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.

3.1 Parameterized Public Announcement Logic

The logic PPAL will be evaluated on a parameterized Kripke structure, 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).
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 instantiation, 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 \( M = (S, AP, \leadsto, L, \cdot) \) 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 \( L_i(s) \subseteq AP; \)
- \( \leadsto \subseteq S \times \mathbb{N} \times S \) is a \( \mathbb{N} \)-labelled accessibility relation between states, called indistinguishability relation, such that any triple \( (s, i, s') \leadsto \) satisfies \( 0 \leq i < |s| = |s'| \). We assume: for any \( s \in S \) and \( 0 \leq i < |s| \), we have \((s, i, s) \leadsto \).

\((s, i, s') \leadsto \) is written \( s \leadsto_i s' \) and reads "if \( s \) is the actual state of the system (world), the \( i \)-th agent observes 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 \( \leadsto_i \) 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 \( S = \{m, c\}^* \), \( AP = \{m, c\} \), and the size \( |w| \) of a state \( w \in S \) is defined as its length. For all \( i \in [|w]| \), \( L_i(w) = \{m\} \) if \( w[i] = m \) and \( \cdot \) otherwise.

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

\[ \varphi ::= \top \mid \varphi \land \varphi \mid \neg \varphi \mid \exists i : \varphi \mid i = 0 \mid i + k \mid p_i \mid (i)\varphi \mid \{\psi\} \varphi \]

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 \( i \)-th agent in the system. This is to some extent akin to how indexed LTL extends LTL [8]. 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 \( k \) (mod \( d \)) for some constants \( k \) and \( d \)). This is essentially the extension of the difference logic [18] 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 \( \lor, \land, \leftrightarrow \) and universal quantification \( \forall \) can be encoded in a standard way. The formula \(|i|\varphi \equiv \neg((i)\neg\varphi)\) encodes that agent \( i \) knows with certainty that \( \varphi \) holds. Usage of constants is also allowed: \( i = k \equiv \exists j : j = 0 \land i = j + k \), \( k \varphi \equiv \exists i : i = k \land p_i \), \( (k)\varphi \equiv \exists i : i = k \land (i)\varphi \).

We denote \( FV(\varphi) \) for the set of ("not quantified") free variables, of \( \varphi \). We say that \( \varphi \) 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 \( \mu \) and a formula \( \varphi \), 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 \( M \) a state \( s \in S \), a PPAL formula \( \varphi \), and a valuation \( \mu \in \mathbb{N}^{FV(\varphi)} \), we define the satisfaction relation \( \models \), inductively, by \( M, s, \mu \models \varphi \) if, and only if, \( \forall i, \mu(i) \in [[s]] \) and one of the following condition holds:

- \( \varphi = \top \)
- \( \varphi = \psi_1 \land \psi_2 \) and \( M, s, \mu \models \psi_2 \)
- \( \varphi = \neg \psi \) and \( M, s, \mu \models \psi \)
- \( \varphi = 3i : \psi_1 \land \psi_2 \) for some \( \mu' \) s.t. \( \forall x \neq i, \mu(x) \equiv \mu'(x) \)
- \( \varphi = i = j + k \) and \( \mu(i) \equiv \mu(j) + k \)
- \( \varphi = i = 0 \) and \( \mu(i) \equiv 0 \)
- \( \varphi = i \varphi \) and \( \forall i, \mu(i) \equiv 0 \)
- \( \varphi = p_i \) and \( p \in L \nu(i) \)
- \( \varphi = (i)\psi \) and there exists \( t \in S \) such that \( s^{\mu(i)} t \land M, t, \mu \models \psi \)
- \( \varphi = \{\psi_1\} \psi_2 \) and \( M, s, \mu \models \psi_1 \implies M(\{\psi(\mu)\})_\mu, s, \mu \models \psi \)

where for any closed PPAL formula \( \psi \), \( M(\{\psi\}) \) is the (parameterized) Kripke structure \( M \) restricted to the state space satisfying \( S(\{\psi\}) = \{s \mid M, s \models \psi \} \).

We 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 \( \varphi \land \{\psi!\} \psi \) 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:

\[ \{\exists i : m_i \land \forall j, i \neq j \rightarrow \neg m_j\} \land i, \{i\}m_i \lor \{i\} \neg m_i \]

### 3.2 Regular Kripke Structures

We now provide a regular presentation of parameterized Kripke structures, and define the model checking problem.
Our main result in this section is the decidability of regular model checking of PPAL.

\[ \varphi(M) := \{ (s, \mu) \in S \times \mathbb{N}^{\mathcal{FV}(\varphi)} \mid M, s, \mu \models \varphi \} \]

Although we are considering non-pointed Kripke structures, the setting is not restrictive here, as initial states could be specified by adding an extra atomic proposition \( init \in AP \) and replacing \( \varphi \) by \( \varphi' \equiv init \rightarrow \varphi \).

## 4 Regular Model Checking of PPAL

Giving a regular Kripke structure \( M \) and a closed PPAL formula \( \varphi \), its semantics \( \llbracket \varphi \rrbracket(M) \) 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. The following lemma makes our claim more precise:

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

\[ \llbracket \varphi \rrbracket(T) = \{ s \otimes \sigma \mid \exists \mu \in \mathbb{N}^X : \sigma = \mathcal{V}(\mu, |s|) \land M_{\sigma}, s \models \varphi \} \]

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

**Proof.**

- \( \llbracket \top \rrbracket(T) = \pi_{(\Sigma, \mathbb{R}, \mathbb{B}, X)}(T) \);
- \( \llbracket \varphi_1 \land \varphi_2 \rrbracket(T) = \llbracket \varphi_1 \rrbracket(T) \cap \llbracket \varphi_2 \rrbracket(T) \);
- \( \llbracket \neg \varphi \rrbracket(T) = \llbracket \top \rrbracket(T) \setminus \llbracket \varphi \rrbracket(T) \);
- An existential quantification over \( i \in X \) is implemented by removing the information about \( i \)'s position. For \( \alpha = t \diamond v \circ x \in ((\Sigma \times \mathbb{B} \times \Sigma) \times \mathbb{B}^X) \), we define \( F(\alpha) = t \omega v[i/x] \).

Hence, \( \exists \alpha : \llbracket \varphi \rrbracket(T) = F(\llbracket \varphi \rrbracket(T) \otimes X^{0^k}) \);

- \( i = j + k \) is encoded by the fixed regular expression:

\[ L_k = \{(0, 0)^* (1, 0) (0, 0)^k (1, 0)^0 \text{ if } k > 0 \}
\]

\[ (0, 0)^* (1, 1) (0, 0)^0 \text{ otherwise} \]

For \( w \otimes v \in (\Sigma \times \mathbb{B} \times \Sigma) \times \mathbb{B}^X \), we consider the morphism \( F \) defined on any tuple \( \alpha = t \circ u \circ (v(i), v(j)) \) by \( F(\alpha) = t \circ u \).

we finally have \( \llbracket \exists i = j + k \rrbracket(T) = F(\llbracket \top \rrbracket(T) \otimes L_k) \);

- \( \llbracket \rho_i \rrbracket(T) = \pi_{(\Sigma, \mathbb{R}, \mathbb{B}, X)}(T) \cap A^* B A^* \), where

\[ A = \{ \alpha \in \Sigma \mid t \not\models L(\alpha) \times (v, v(i)) = 0 \}
\]

\[ B = \{ \alpha \in \Sigma \mid t \not\models L(\alpha) \times (v, v(i)) = 1 \}
\]

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 \( \varphi \).

- 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 \( \sigma \), the regular Kripke structure obtained from \( M_v \), after announcing \( \varphi(\mu_v) \):

\[ T(\varphi!) = \bigcup_{\sigma \in \mathbb{B}^X} \left( T_{M_v}(\varphi(\mu_v)) \right) \otimes \{ \sigma \} \]

\( T(\varphi!) \) is actually regular: we first build \( \llbracket \varphi \rrbracket(T) \) in order to construct a regular Kripke on this state space. In order
Moreover, this translation does not apply to the newly introduced blow-ups, so the overall running time may become non-elementary.

\[ T(\varphi) = T \land F(\psi)(T) \times 0^* \times F(\psi)(T). \]

Finally, we conclude with the implementation of the (vacuous truth) semantics of the public announcement:

\[ \begin{array}{c}
\{ \varphi \}' \psi \equiv (\neg \varphi) (T) \cup (\psi)(T(\varphi)) \\
\end{array} \]

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 \( M \) 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 \( \{ \forall i, (\neg) - m_i! \} \), it is further reduced to \( \Sigma^*(\{ m \}) \Sigma^*(\{ m \}) \). This sequence of announcements, however, cannot continue forever as each 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 \( \text{iterated public announcement} \) operator [22];

\[ \{ \varphi \}, \{ \varphi \}, \ldots, \{ \varphi \} \]

arbitrarily many times.

Definition 4.4. A formula \( \varphi \) is in PPAL* if it is in the grammar of PPAL, augmented with \( \varphi ::= \{ \varphi \} \{ \varphi \}^* \varphi \) with \( \varphi \in \Sigma \cap \mathbb{N} \). The semantics is given by induction on \( k \):

\[ \begin{array}{l}
\{ \varphi \}^0 \psi (M) = \psi (M); \\
\{ \varphi \}^{k+1} \psi (M) = \{ \varphi \}^k (\{ \varphi \} \psi ) (M); \\
\{ \varphi \}^* \psi (M) = \bigcup_{k \geq 0} \{ \varphi \}^k \psi (M). \\
\end{array} \]

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 blow-ups, so the overall running time may become non-elementary.

(2) We construct now a regular Kripke structure \( M \) such that its PPAL* theory is undecidable. To this end, we encode the Minsky machine halting problem [23]: a 2-counter (Minsky) machine is a tuple \( (Q, q_0, q_f, \delta) \) where

- \( Q \) is a finite subset;
- \( q_0 \in Q \) is the initial state;
- \( q_f \in Q \) is the final state;
- \( \delta \subseteq Q \times \{ \text{test, inc, dec} \} \times \Sigma \times Q \) is the set of transitions.

The semantics of such a machine is defined over the configuration space \( Q \times \mathbb{N}^2 \), with \( (q, x_1, x_2) \xrightarrow{t} (r, y_1, y_2) \) if, and only if, the following conditions hold:

\[ t = (q, o, i, r) \in \delta \text{ for some } (o, i) \in \{ \text{test, pos, inc, dec} \} \times \Sigma; \]

\[ x_{i-1} = y_{1-i}; \]

\[ \text{if } o = \text{test, then } x_i = y_i = 0; \]

\[ \text{if } o = \text{pos, then } x_i = y_i > 0; \]

\[ \text{if } o = \text{inc, then } x_i + 1 = y_i; \]

\[ \text{if } o = \text{dec, then } x_i + 1 = y_i + 1; \]

We say that such a machine terminates if there exists a finite path from configuration \((q_0, 0, 0)\) to configuration \((q_f, x_0, x_1)\) for some \((x_0, x_1)\).

Moreover, we assume, without loss of generality, our 2-counter machines to be deterministic, namely: if for any configuration \( y \), there exists at most one \( t \in \delta \) such that \( y \xrightarrow{t} y' \) for some \( y' \).

We consider the regular Kripke structure \( M \) with \( AP = \{ p, c^{(1)}, c^{(2)} \} \), and \( T_M \) the complete transducer \( \Sigma^* \otimes 0^* \otimes \Sigma^* \).

Given a 2-counter machine \((Q, q_0, q_f, \delta)\), we construct a formula \( \varphi \) such that the machine terminates if, and only if, \( \{ \varphi \} (M) \neq \emptyset \). We assume for our encoding that \( Q = \{ [0] \} \), allowing us to encode the current state as a unary position. Let’s first restrict the model to states where each proposition is true at exactly one position, by announcing:

\[ \varphi_m = \exists \! i_0, i_1, i_2 : \left\{ \begin{array}{l}
\forall j, c_j^{(0)} \rightarrow i_0 = j \land c_j^{(1)} \rightarrow i_1 = j \land p_j \rightarrow i_2 = 0 \\
\forall j, c_j^{(0)} \rightarrow i_0 = j \land c_j^{(1)} \rightarrow i_1 = j \land p_j \rightarrow i_2 = 0 \\
\end{array} \right. \]

Intuitively, a configuration \((q, x_0, x_1)\) will be encoded as the state word \( V(q_0, l) \otimes V(x_0, x_1) \otimes V(q, l) \) for any \( l \geq \max(q, x_0, x_1) \).

We construct now a formula \( \varphi_t \) expressing that the current configuration still has successor:

\[ \varphi_t = \exists \! i_0, i_1, j \neq q_f : c^{(0)}_{i_0} \land c^{(1)}_{i_1} \land \bigvee_j \left( j = q \lor \langle 0 \rangle (p_q' \land \varphi_{\text{op}, k}) \right) \]

where

\[ \varphi_{\text{op}, k} = \left\{ \begin{array}{l} i_k = 0 \land c_i^{(0)} \land c_i^{(1)} \text{ when } o = \text{test} \\
\exists l : i_k = i + 1 \land c_i^{(0)} = c_i^{(1)} \text{ when } o = \text{pos} \\
\forall l, l = i + 1 \land c_i^{(0)} \land c_i^{(1)} \text{ when } o = \text{dec} \\
\end{array} \right. \]

Note that a state \( s \) encoding the configuration \((q, x_0, x_1)\), where a increment of \( x_1 \) is available but \( x_2 = |s| - 1 \), is never removed, even though state \( s \) has no successor in the current Kripke structure; we adopt this convention since any state in \( (0^* \cdot s) \) still has a successor.
We encode now the whole formula:
\[ \varphi = p_0 \land (a_0^{(0)} \land (\{a_m^1\} \land (q_f))^* \uparrow \bot) \]
We claim that \( \llbracket \varphi \rrbracket (M) \neq \emptyset \) if, and only if, the machine terminates.

- If the machine terminates, there exists a finite run 
  \((q_0, x_0, y_0) \ldots (q_f, x_n, y_n)\) with \( q_0 = q_f \) and \( x_0 = y_0 = 0 \).
  We let \( l = \max (x_i, y_i \mid 0 \leq i \leq n) \). We prove by (decreasing) induction on \( i \) that \( V(x_i, l) \otimes V(y_i, l) \otimes V(q_i, l) \in (\llbracket \{q_m^1\} \land (q_f))^* \uparrow \bot) (M) \).
  The result holds for \( i = n \) since \( q_f \) is not satisfied for \( q_f \); then, we follow the semantics definition for the induction case: by determinacy of the machine, there exists at most one valid instruction \((q_{i-1}, q_i, y_i)\).
  Because no state can satisfy \( \bot \), the state \( V(x_i, l) \otimes V(y_i, l) \otimes V(q_i, l) \) has to eventually be removed, hence \( q_i \) becomes eventually unsatisfied from \( V(x_i, l) \otimes V(y_i, l) \otimes V(q_i, l) \).
- For the converse implication, assume the machine has an infinite run denoted with \((q_i, x_i, y_i)\). For a fixed \( i \in \mathbb{N} \), we check by induction on \( k \), for all such \( i \), that \( l = \max (x_i, y_i, q_i) \), we still have \( M \cup V(x_i, l) \otimes V(y_i, l) \otimes V(q_i, l) \neq (\llbracket \{q_m^1\} \land (q_f))^* \uparrow \bot) \).
  - For \( k = 0 \), the formula reduces to \( (\llbracket \{q_m^1\} \land (q_f))^* \uparrow \bot) \), and since all states are proper encoding of configurations, they satisfy \( \varphi \) but not \( \bot \).
  - Assuming the result is true for some \( k \in \mathbb{N} \), we prove the result still holds for \((q_i, x_i, y_i)\) at step \( k + 1 \), by applying either applying the induction hypothesis on \((q_{i+1}, x_{i+1}, y_{i+1})\) at step \( k \), or checking that \( x_{i+1} > l \) and \( y_{i+1} \) meaning \( V(x_i, l) \otimes V(y_i, l) \otimes V(q_i, l) \) satisfies \( q_i \) through its increment clause.
  That is to say: \( \llbracket \varphi \rrbracket (M) \land \Sigma^l \neq \emptyset \). Since the result holds for any \( i \in \mathbb{N} \), we conclude that \( \llbracket \varphi \rrbracket (M) = \emptyset \).

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 \( \{q_f\}^* \uparrow \bot \), whose semantics can be seen as the set of states not being removed, after arbitrarily many state space restriction operated by the public announcement \( \{q_f\} \).

For the rest of this section, we fix a more general setting with:
- An alphabet \( \Sigma \);
- An initial state space \( \mathcal{S} \subseteq \Sigma^* \);
- A function \( F : 2^\Sigma \rightarrow 2^\Sigma \) restricting the state space, that is to say: \( \forall X \subseteq \Sigma^*, F(X) \subseteq X \).
- For all \( k \in \mathbb{N} \), we let \( \mathcal{S}_{k+1} = F(\mathcal{S}_k) \) and \( \mathcal{S}_\infty = \cap_{k \geq 0} \mathcal{S}_k \).

Moreover, we assume \( \mathcal{S} \) 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 \( \mathcal{S}_\infty \). 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 4.3, or the following counterexample.

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

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

5.1 Unique Characterization

Let us first remark that the application \( F \) is not necessarily monotone. Consider for example the announcement \( \forall i, m_i \forall [t] \rightarrow m_i \) which reads:

“every non-muddy child knows he’s not muddy.”

Then \( F([cm]) = [cm] \) but \( F([cm, mm]) = \emptyset \). As a consequence, \( \mathcal{S}_\infty \) 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 \( \leq \) is defined for every \((s, t) \in \mathcal{S}^2 \), by:

\( s \leq t \) if, and only if, \( \forall k \in \mathbb{N}, s \in \mathcal{S}_k \Rightarrow t \in \mathcal{S}_k \).

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

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

Definition 5.3. For a relation \( R \subseteq \mathcal{S} \times \mathcal{S} \), and any \( s \in \mathcal{S} \), we define the upward-closure and equivalence class of \( s \) by \( \uparrow_R s = \{u \in \mathcal{S} \mid (s, u) \in R \} \), and \( [s]_R = \{u \in \mathcal{S} \mid (u, s) \in R \} \).

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

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

Proposition 5.4. Let \( R \subseteq \mathcal{S} \times \mathcal{S} \). If \( R = \leq \), then:

(1) \( R \) is a total pre-order on \( \mathcal{S} \), and
(2) \( \forall s \in \mathcal{S}, \forall s \in \mathcal{S} \rightarrow \uparrow_R s \land [s]_R = \uparrow_R s = F(\uparrow_R s) \).

Moreover, the converse holds whenever \( \mathcal{S} \) is finite.

Sketch. A proof of the direct implication being already sketched above, we focus on the converse implication:

Assuming that \( R \) satisfies the above conditions and \( \mathcal{S} \) is finite, we prove that \( R = \leq \), by induction on \( |\mathcal{S}| \).

The result is trivial when \( |\mathcal{S}| = 0 \), consider now \( \mathcal{S} \neq \emptyset \) and some minimal \( s \in \mathcal{S} \) with respect to the total pre-order \( R \), that is to say \( \uparrow_R s = \mathcal{S} \).
• If \([s, t]_R = \{s, t\} \cap (\{s\} \cup \{t\})\), then \([s, t]_R = \emptyset\). Consider \(\leq^\ast\). \(\leq^\ast\) is the disappearance relation of \(F\), with initial set \(S_0\), and \(R^\ast = R \cap S_0 \times S_0\). We easily check that \(\leq^\ast\) is the disappearance relation on \(S_0\), such that for every \(s, t \in S_0\), \([s, t]_R = \emptyset\) and \(R = R \cap S_0 \times S_0\). We can therefore apply the induction hypothesis on \(S_0\): \(\leq^\ast\) and \(R\) coincide on \(S_0 \times S_1\).

5.2 Learning Procedure

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

\[(R_1): \text{"} F\text{"} \text{is length-preserving} \]

Strictly speaking, we assume \(L_\ast\) to be the representation of the family \((\leq_k)_{k \in \mathbb{N}}\), where for each \(k\), \(\leq_k\) is the disappearance relation starting from the initial state space \(S_0 \cap \Sigma^k\). As \(\Sigma^k\) is finite for a given \(k\), this restriction further allows us to provide a unique characterization of \(L_\ast\), as provided by Proposition 5.4.

We now introduce the \(L_\ast\) algorithm from Angluin, which allows us to learn a finite automaton \(A\), or equivalently a target regular language \(L_\ast \in \text{Reg}(\Sigma_\ast)\), based on queries answered by an Oracle. Such an Oracle has to answer so-called membership and equivalence queries, either by having direct access to the target language or by indirect means.

We explain the exact semantics of \(L_\ast\) queries for a target language language \(L_\ast \in \text{Reg}(\Sigma_\ast)\), and how they are answered in this learning procedure, where the target language is \(L_\ast \subseteq (\Sigma \times \Sigma)^k\):

• **Membership Queries:** The Oracle is asked whether a given word \(w \in \Sigma^*\) is in the target language \(L_\ast\).

**Answer:** We let \(s, t \in \Sigma^{|w|}\) with \(s \otimes t = w\) and decide whether \(s \leq^\ast t\). We proceed to the iterative computation of the sets \((\leq_k)_{k \in \mathbb{N}}\) and stop whenever \(s \otimes t = w\) 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, \(s, t \in S_\infty\). To circumvent this issue, we perform the computation on the restricted state space of a fixed length \(|w|\), namely \(S_k \cap \Sigma^{|w|}\), ensuring a finite cardinality. As soon as \(s, t \in S_k \cap \Sigma^{|w|}\), we conclude that \(s \leq^\ast t\). This leads to our second requirement:

\[(R_1): \text{"} F\text{"} \text{restricts the state space independently for different state sizes."} \]

• **Equivalence Queries:** Given a candidate language \(L_\ast\), the Oracle is asked whether \(L_\ast = L_\ast\) and if not, provides a counterexample \(w \in L_\ast \cup L_\ast\).

**Answer:** We make use of Proposition 5.4, which can be seen as a first order characterization of \(\leq^\ast\), 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 counter example for \(L_\ast\). For example, a counter example for the transitivity property would consist in a triple \((s_1, s_2, s_3) \in S^3\) such that \(s_1 \otimes s_2 \in L, s_2 \otimes s_3 \in L\) or \(s_1 \otimes s_3 \in L\), 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 \(l\), we are guaranteed that \(L_\ast\) restricted to \((\Sigma \times \Sigma)^{\leq l}\) is not a proper encoding of \(\leq \cap S^3\). A direct enumeration of the sequence \((\Sigma_k \cap \Sigma^l)_{k \geq 0}\) will therefore terminate and therefore will 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 formulæ.

**Proposition 5.5.** Let \(R \subseteq \Sigma^\ast \times \Sigma^\ast\) and \(k \in \mathbb{N}\).

\(R \cap (\Sigma^k \times \Sigma^k) \neq \emptyset\) if, and only if, any one of the conditions holds:

1. \(\exists s, t : (s, t) \in R \land (s, t) \notin S \times S\);
2. \(\exists s : (s, s) \notin R\);
3. \(\exists s_1, s_2, s_3 : (s_1, s_2) \in R \land (s_2, s_3) \in R \land (s_1, s_3) \notin R\);
4. \(\exists s, t : (s, t) \notin R \land ((s, t) \notin R\)

\[\exists s, t_1, t_2 : \begin{cases} (s, t_1) \in R \land (s, t_2) \in R \\ (t_1, s) \notin R \iff t \in F(\{t\}) \\ (t_2, s) \notin R \lor t \notin F(\{t\}) \end{cases}\]

Where all quantifications are made over \(\Sigma^k\).

**Proof.** Property (1) enforces \(R \subseteq \Sigma^\ast \times \Sigma^\ast\) while properties (2)–(4) encode respectively reflexivity, transitivity and totality, as stated by Proposition 5.4, after taking the negation.

We provide here a proof of property (5) built on top of the second property not being fulfilled. Recall first that \(F(X) \subseteq X\) for all \(X\), and \(|s|_R \leq |r|_R^\ast\) for any \(s\), hence condition \([s]_R = \{s, t\} = F(\{t\})\) is equivalent to \(\{s, t\} \subseteq [s]_R \cap F(\{t\})\).

After taking the negation, the second property of Proposition 5.4 becomes: \(\exists s : [s]_R \notin F(\{t\}) \land \{t\} \notin [s]_R \cap F(\{t\})\) which is equivalent to:

\[\exists s, t_1, t_2 : \begin{cases} ((s, t_1) \in R \land (t_1, s) \in R) \iff ((s, t_1) \in R \land t_1 \notin F(\{t\})) \\ (s, t_2) \in R \land ((t_2, s) \notin R \lor t \notin F(\{t\})) \end{cases}\]

Hence, after factorizing by \((s, t_1) \in R\):

\[\exists s, t_1, t_2 : \begin{cases} (s, t_1) \in R \land ((t_1, s) \in R \iff t_1 \notin F(\{t\})) \\ (s, t_2) \in R \land ((t_2, s) \notin R \lor t \notin F(\{t\})) \end{cases}\]
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 $L \cap \Sigma \otimes \Sigma \subseteq \emptyset$.

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

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

This condition is introduced as the last requirement:

(53): “$F$ is effective and uniformly regular”

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

**Definition 5.6.** Let $G$ be a function from $2^\Sigma_i$ to $2^\Sigma_i$.

- $G$ is independently length-preserving if:
  \[\forall l \in \mathbb{N} \forall \xi \subseteq \Sigma_i, G(X \cap \Sigma_j^l) = G(X) \cap \Sigma_j^l;\]
- $G$ is effectively uniformly regular if:
  For any given alphabet $\Sigma'$ and $L \in \text{Reg}(\Sigma' \times \Sigma_1)$, the following language is regular and computable:

$$\left\{w' \otimes w_2 \mid \exists w_1 \in \Sigma_i^{\left\lfloor \frac{l}{2} \right\rfloor} : w_2 \in G(\{w_1 \mid w' \otimes w_1 \in L\})\right\}$$

**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 $L \leq$ if, and only, it is regular.

**Proof.** Thanks to the length-preserving property of $F$, the relation $\leq_k = \Sigma^k \times \Sigma^k \cap \leq$ coincide with the disappearance relation initiated from $\emptyset_0 \cap \Sigma^k$ with the same operator $F$.

Uniform and effective regularity enables the effective implementation of the Oracles:

- First of all, membership queries, as well as counterexample generation in the equivalence queries, require the computation of sequences $F_i(\Sigma \times \Sigma)$ for different values of $i, l \in \mathbb{N}$. This can be seen as a weaker form of effective regularity, satisfied by the operator.
- Equivalence queries implementation relies on the translation of the conditions provided by Proposition 5.5 into regular queries on transducers over $\Sigma \times \Sigma$. Last condition in particular, requires, for a candidate relation $R$, encoded as a transducer $R \in \text{Reg}(\Sigma \times \Sigma)$, the computation of

$$\left\{s \otimes t \mid t \in F(\uparrow_R s)\right\} = \left\{s \otimes t \mid \exists u \in \Sigma_i^{\left\lfloor \frac{l}{2} \right\rfloor} : t \in F(\{u \mid s \otimes u \in L\})\right\}$$

We conclude with the termination guarantees:

- If $L \leq$ is regular, the $L^*$ procedure terminates in polynomial time[5].
- Conversely, if the procedure terminates, the returned language $L$ is regular and passed the equivalence query. Therefore, it satisfies the $L \leq$ characterization provided by Proposition 5.5, so $L \leq = L$ is regular.

### 5.4 Application to PPAL*

We finally address the general case with the following observation:

**Proposition 5.8.** Let $\varphi$ and $\psi$ be two closed formula and $M$ a parameterized Kripke structure, whose state space is $\Sigma$. For any $X \subseteq \Sigma$, we define $M|_X$ for the parameterized Kripke structure restricted to $X$ and consider the resulting disappearance relation $\leq_X \subseteq \Sigma^2$.

We have:

$$\Leftrightarrow (M) = \{s \in \Sigma \mid \exists t \in \Sigma : [[\varphi]](M|_X)\}$$

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

**Proposition 5.9.** Let $\varphi$ be a closed formula on a regular Kripke structure $M$. The application, $F_\varphi$ defined by

$$\forall X \subseteq \Sigma, F_\varphi(X) = [[\varphi]](M|_X)$$

is length-preserving, effectively and uniformly regular.

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

### 6 EXPERIMENTS

We developed a prototype tool implementation, using the Java libraries Learnlib and Automatalib [14]. Three different models were specified then verified showing tractability of the procedure:

| Model                | Duration | Memory Usage |
|----------------------|----------|--------------|
| Russian cards        | 36s      | 2365MB       |
| Large number         | 53s      | 1218MB       |
| M $\leq$ 3 Muddy children | 3s      | 130Mo        |
| M $\leq$ 4 Muddy children | 3s      | 162Mo        |
| M $\leq$ 9 Muddy children | 24s     | 1136Mo       |
| M $\leq$ 10 Muddy children | TO (5min+) |           |
| M $\leq$ 11 Muddy children | out of memory | |
| 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 $M$, and for each specification $\varphi$, computes its satisfaction set. In case the complement $[[\neg \varphi]](M)$ is non-empty, an NFA is returned, which can be interpreted as the set of counterexamples to $\varphi$. 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.

**Automaton size.** Since specifying a transducer for $\rightarrow$ 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

---

3Experiments were conducted on a i7-8550U CPU @ 1.80GHz machine with 16GB of RAM and JavaSE-1.8. The prototype and models are available online [28].
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, 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 [33] 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 are a = 1, b = 2, and c = 3. For x ∈ AP and i ∈ N, xi holds iff agent x has card i in their hand. Moreover, we assume that ai, bi and ci are mutually exclusive (each card appears only in one hand). We easily check that 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}, \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:

\[ \psi \equiv i \land \psi ! \]  

// truthful PA
\[ \forall i, (b) a_i \lor (b) b_i \lor (b) c_i \]  

// b knows the distribution
\[ \forall i, \neg c_i \rightarrow (c) a_i \land (c) \neg a_i \]  

// c doesn’t know
\[ (c) b_i \land (c) \neg b_i \]  


While [3] provides several sufficient and necessary conditions on the number of cards received by each participant, we focus here on a single example of (sufficent) good announcement, provided by [3, Proposition 5] in the case where N % 3 = 0, Alice receives 3 cards, Cathy only one, and Bob the rest (property \( \varphi_{\text{model}} \)).

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

\[ \psi \equiv \exists j : j \mod 3 = 0 \land (a_j \land a_{j+1} \land a_{j+2}) \lor (a_j \land a_{j+4} \land a_{j+6}) \]

Which can be checked with the verification question:

\[ M \models \varphi_{\text{model}} \]  

We leave to the reader the generalization to any initial hand of Alice and the specification of \( \varphi_{\text{model}} \).

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 \( \varphi \) and encode the observation of \( \varphi \) for the formula where any propositional sub-formula \( p_{i} \) has been rewritten into \( i = x \land a_{y} \lor i = y \land a_{x} \lor i \not\in \{x, y\} \land a_{i} \). This protocol converges 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.

Remark: If Alice is not given the cards 0,1,2, nor another combination specified by \( \varphi \), the announcement is not valid. However, it can be rewritten, depending on Alice’s current hand. Let x and y be two index variables. For any \( \varphi \) formula, we denote \( \varphi[x/y] \) for the formula where any propositional sub-formula \( p_{i} \) has been rewritten into \( i = x \land a_{y} \lor i = y \land a_{x} \lor i \not\in \{x, y\} \land a_{i} \). The previous verification question is converted into:

\[ M \models \forall \varphi_{\text{model}} \]  

Note that the choice of a satisfying set of indices x1, x2, x3, y1, y2 must not be serendipity: it should work for all possible hands of b and c, that a may imagine, hence the universal \[ a \] quantification. Note also that we need to swap only two pairs of cards, to reconstitute a triple of cards appearing in \( \varphi \).

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 announcement made on the regular Kripke structure of Example 3.8.

\[ \{ i : m_{i} \} \]  

Intuitively, the effect of this announcements is to construct the product automaton of the original transducer \( T_{M} \) 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 \( \forall i, (i = m_{i})^{*} \), which reduces to the disappearance relation computation: for \( s, t \in S_{0}, s \leq t \) if, and only if, \( |s|_{m} \leq |t|_{m} \). As a matter of fact, the protocol
terminated after $|s|_m$ announcements of the father whichever 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 $M$. As predicted, our algorithm successfully computes a transducer for $\leq$, with $O(M)$ states.

6.4 Unbounded Case and Symmetry Reduction

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

Nonetheless, we observe that the problem is invariant under permutation, more precisely:

- The formula $\varphi$ lies in a fragment of PPAL$^*$ without index comparison of the form $i = j + k$ for any $k \neq 0$;
- For any word $w \in T_M$ and any bijection $\Sigma$ on $[[w]]$, $w(\sigma(0)) \ldots w(\sigma(|w| - 1)) \in T_M$.

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 $s \rightarrow t$ 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 $c^n m^p$, our rewriting actually consists in a unary encoding of the number clean and muddy children. As for the largest number challenge, the disappearance relation is regular, and we successfully verify the rewritten formula:

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

7 RELATED AND FUTURE WORK

Related Work. Finite-state model checkers for various epistemic logics are available, e.g., MCMAS [21], DEMO [32, 37], SMCGEL [31], and MCK [12]. Kouvaros and Lomuscio [17] have studied cutoff techniques for ACTL$^* \setminus X$, a temporal-epistemic logic combining S5 and temporal logic ACTL$^* \setminus X$, 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 [8, 38]) — 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 [33], and the large number challenge, all of which have been studied in the finite-state case (e.g. see [21, 31, 32, 37]).

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 [21]. 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 [19] with transducer techniques, a lossless Parikh image [24] could be computed in terms of a Presburger formula [26]. As the PPAL semantics involves only boolean, synchronous product and morphism operations, the computation could be performed in this domain. We leave this for future work. We would also like to investigate the possibility of developing cutoff methods of [17] for the examples that we consider in this paper.

ACKNOWLEDGMENTS

This work was supported by the ERC Starting Grant 759969 (AV-SMP) 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, 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
[3] 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
[4] Benjamin Aminof, Aniello Murru, 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?doid=2937098
[5] Dana Angluin. 1987. Learning Regular Sets from Queries and Counterexamples. Inf. Comput. 75, 2 (Nov. 1987), 87–106. https://doi.org/10.1016/0019-9958(87)90052-6
[6] 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
[7] 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
[8] 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/S00656ED1V01Y201505DCTI0
[9] A. Blumensath. 1999. Automatic Structures. Master’s thesis. RWTH Aachen.
[10] A. Blumensath and E. Grädel. 2004. Finite Presentations of Infinite Structures: Automata and Interpretations. Theory Comput. Syst. 37, 6 (2004), 641–674.
[11] Ronald Fagin, Joseph Y. Halpern, Yoram Moses, and Moshe Vardi. 1995. Reasoning about Knowledge. MIT Press.
[12] 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
[13] 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
[14] Malte Isberner, Falk Howar, and Bernhard Steffen. 2015. The Open-Source LearnLib. In Computer Aided Verification, Daniel Kroening and Corina S. Pásztor (Eds.). Springer International Publishing, Cham, 487–495.
[15] Michael J. Kearns and Umesh V. Vazirani. 1994. *Computational Learning Theory*. MIT Press. https://mitpress.mit.edu/books/introduction-computational-learning-theory
