# Probabilistic Bisimulation for Parameterized Systems
## (with applications to verifying anonymous protocols)*

Chih-Duo Hong[1], Anthony W. Lin[2], Rupak Majumdar[3], and Philipp Rümmer[2]

[1] Oxford University, United Kingdom
[2] TU Kaiserslautern, Germany
[3] Uppsala University, Sweden
[4] Max Planck Institute for Software Systems, Germany

**Abstract.** Probabilistic bisimulation is a fundamental notion of process equivalence for probabilistic systems. It has important applications, including the formalisation of the anonymity property of several communication protocols. While there is a large body of work on verifying probabilistic bisimulation for finite systems, the problem is in general undecidable for parameterized systems, i.e., for infinite families of finite systems with an arbitrary number $n$ of processes. In this paper we provide a general framework for reasoning about probabilistic bisimulation for parameterized systems. Our approach is in the spirit of software verification, wherein we encode proof rules for probabilistic bisimulation and use a decidable first-order theory to specify systems and candidate bisimulation relations, which can then be checked automatically against the proof rules.
We work in the framework of regular model checking, and specify an infinite-state system as a regular relation described by a first-order formula over a universal automatic structure, i.e., a logical theory over the string domain. For probabilistic systems, we show how probability values (as well as the required operations) can be encoded naturally in the logic. Our main result is that one can specify the verification condition of whether a given regular binary relation is a probabilistic bisimulation as a regular relation. Since the first-order theory of the universal automatic structure is decidable, we obtain an effective method for verifying probabilistic bisimulation for infinite-state systems, given a regular relation as a candidate proof. As a case study, we show that our framework is sufficiently expressive for proving the anonymity property of the parameterized dining cryptographers protocol and the parameterized grades protocol. Both of these protocols hitherto could not be verified by existing automatic methods. Moreover, with the help of standard automata learning algorithms, we show that the candidate relations can be synthesized fully automatically, making the verification fully automated.

## 1 Introduction

Equivalence checking using bisimulation relations plays a fundamental role in formal verification. Bisimulation is the basis for substitutability of systems: if two systems

* This research was sponsored in part by the ERC Starting Grant 759969 (AV-SMP), ERC Synergy project 610150 (ImPACT), the DFG project 389792660-TRR 248 (Perspicuous Computing), the Swedish Research Council (VR) under grant 2018-04727, and by the Swedish Foundation for Strategic Research (SSF) under the project WebSec (Ref. RIT17-0011).

are bisimilar, their behaviors are the same and they satisfy the same formulas in expressive temporal logics. The notion of bisimulation is defined both for deterministic [39] and for probabilistic transition systems [34]. In both contexts, checking bisimulation has many applications, such as proving correctness of anonymous communication protocols [15], reasoning about knowledge [22], program optimization [32], and optimizations for computational problems (e.g. language equivalence and minimization) of finite automata [12].

The problem of checking bisimilarity of two given systems has been widely studied. It is decidable in polynomial-time for both probabilistic and non-probabilistic *finite-state* systems [6, 17, 20, 52]. These algorithms form the basis of practical tools for checking bisimulation. For infinite-state systems, such as parameterized versions of communication protocols (i.e. infinite families of finite-state systems with an arbitrary number $n$ of processes), the problem is undecidable in general. Most research hitherto has focused on identifying decidable subcases (e.g. strong bisimulations for pushdown systems for probabilistic and non-probabilistic cases [25, 47, 48]), rather than on providing tool support for practical problems.

In this paper, we propose a first-order verification approach—inspired by software verification techniques—for reasoning about bisimilarity for infinite-state systems. In our approach, we provide first-order logic *proof rules* to determine if a given binary relation is a bisimulation. To this end, we must find an *encoding* of systems and relations and a *decidable first-order theory* that can formalize the system, the property, and the proof rules. We propose to use the decidable first-order theory of the *universal automatic structure* [8, 10]. Informally, the domain of the theory is a set of words over a finite alphabet $\Sigma$, and it captures the first-order theory of the infinite $|\Sigma|$-ary tree with a relation that relates strings of the same level. The theory can express precisely the class of all *regular relations* [8] (a.k.a. automatic relations [10]), which are relations $\varphi(x_1, \ldots, x_k)$ over strings $\Sigma^*$ that can be recognized by synchronous multi-tape automata. It is also sufficiently powerful to capture many classes of non-probabilistic infinite-state systems and regular model checking [3, 13, 49–51].

We demonstrate the effectiveness of the approach by encoding and automatically verifying some challenging examples from the literature of parameterized systems in our logic: the anonymity property of the parameterized dining cryptographers protocol [16] and the grades protocol [29]. These examples were only automatically verified for some fixed parameters using finite-state model checkers or equivalence checkers (e.g. see [28, 29]). Just as invariant verification for software separates out the proof rules (verification conditions in a decidable logic) from the synthesis of invariants, we separate out proof rules for bisimulation from the synthesis of bisimulation relations. We demonstrate how recent developments in generating and refining candidate proofs as automata (e.g. [18, 26, 27, 37, 38, 40, 41, 53]) can be used to automate the search of proofs, making our verification fully "push button."

**Contributions.** Our contributions are as follows. First, we show how probabilistic infinite-state systems can be faithfully encoded in the first-order theory of universal automatic structure. In the past, the theory has been used to reason about qualitative liveness of weakly-finite MDPs (e.g. see [36, 37]), which allows the authors to disregard the actual non-zero probability values. To the best of our knowledge, no encoding

of probabilistic transition systems in the theory was available. In order to be able to effectively encode probabilistic systems, our theory should typically be two-sorted: one sort for encoding the configurations, and the other for encoding the probability values. We show how both sorts (and the operations required for the sorts) can be encoded in the universal automatic structure, which requires only the domain of strings. In the sequel, such transition systems will be called *regular transition systems*.

Second, using the *minimal probability assumption* on the transition systems [34] (i.e. there exists an $\varepsilon > 0$ such that any non-zero transition probability is at least $\varepsilon$)—which is often satisfied in practice—we show how the verification condition of whether a given regular binary relation is a probabilistic bisimulation can be encoded in the theory. The decidability of the first-order theory over the universal automatic structure gives us an effective means of checking probabilistic bisimulation for regular transition systems. In fact, the theory can be easily reduced to the weak monadic theory WS1S of one successor (therefore, allowing highly optimized tools like Mona [31] and Gaston [23]) by interpreting finite words as finite sets (e.g. see [19, 46]).

Our framework requires the encoding of the systems and the proofs in the first-order theory of the universal automatic structure. Which interesting examples can it capture? Our third contribution is to provide two examples from the literature of parameterized verification: the anonymity property of the parameterized dining cryptographers protocol [16] and of the parameterized grades protocol [29]. We study two versions of dining cryptographers protocol in this paper: the classical version where the secrets are single bits, and a generalized version where the secrets are bit-vectors of arbitrary length.

Thus far, our framework requires a candidate proof to be supplied by the user. Our final contribution is to demonstrate how standard techniques from the synthesis literature (e.g. automata learning [18, 26, 27, 37, 38, 40, 41, 53]) can be used to fully automate the proof search. Using automata learning, we successfully pinpoint regular proofs for the anonymity property of the three protocols: the two dining cryptographers protocols are verified in 6 and 28 seconds, respectively, and the grades protocol in 35 seconds.

**Other related work.** The verification framework we use in this paper can be construed as a regular model checking [3] framework using regular relations. The framework uses first-order logic as the language, which makes it convenient to express many verification conditions (as is well-known from first-order theorem proving [14]). The use of the universal automatic structure allows us to express two different sorts (configurations and probability values) in one sort (i.e. strings). Most work in regular model checking focuses on safety and liveness properties (e.g. [2, 3, 11, 13, 27, 36, 37, 40, 42, 49, 51, 53]).

Some automated techniques can prove the anonymity property of the dining cryptographers protocol and the grades protocol in the finite case, e.g., the PRISM model checker [28, 45] and language equivalence by the tool APEX [29]. To the best of our knowledge, our method is the first automated technique proving the anonymity property of the protocols in the parameterized case.

Our work is in spirit of deductive software verification (e.g., [4, 14, 24, 35, 43, 44]), where one provides inductive invariants manually, and a tool automatically checks correctness of the candidate invariants. In theory, our result yields a fully-automatic procedure by enumerating all candidate regular proofs, and at the same time enumerating all candidate counterexamples (note that we avoid undecidability by restricting attention to

proofs encodeable as regular relations). In our implementation, we use recent advances in automata-learning based synthesis to efficiently encode the search [18, 37].

## 2 Preliminaries

**General notation.** We use $\mathbb{N}$ to denote non-negative integers. Given $a, b \in \mathbb{R}$, we use a standard notation $[a, b] := \{c \in \mathbb{R} : a \leq c \leq b\}$ to denote real intervals. Given a set $S$, we use $S^*$ to denote the set of all finite sequences of elements from $S$. The set $S^*$ always includes the empty sequence which we denote by $\varepsilon$. We call a function $f : S \to [0, 1]$ a *probability distribution over* $S$ if $\sum_{s \in S} f(s) = 1$. We shall use $I_s$ to denote the probability distribution $f$ with $f(s) = 1$, and $\mathcal{D}_S$ to denote the set of probability distributions over $S$. Given a function $f : X_1 \times \cdots \times X_n \to Y$, the *graph* of $f$ is the relation $\{(x_1, ..., x_n, f(x_1, ..., x_n)) : \forall i \in \{1, \ldots, n\}.\ x_i \in X_i\}$. Whenever a relation $R$ is an equivalence relation over set $S$, we use $S/R$ to denote the set of equivalence classes created by $R$. Depending on the context, we may use $p\, R\, q$ or $R(p, q)$ to denote $(p, q) \in R$.

**Words and automata.** We assume basic familiarity with word automata. Fix a finite alphabet $\Sigma$. For each finite word $w := w_1 \ldots w_n \in \Sigma^*$, we write $w[i, j]$, where $1 \leq i \leq j \leq n$, to denote the segment $w_i \ldots w_j$. Given an automaton $\mathcal{A} := (\Sigma, Q, \delta, q_0, F)$, a run of $\mathcal{A}$ on $w$ is a function $\rho : \{0, \ldots, n\} \to Q$ with $\rho(0) = q_0$ that obeys the transition relation $\delta$. We may also denote the run $\rho$ by the word $\rho(0) \cdots \rho(n)$ over the alphabet $Q$. The run $\rho$ is said to be *accepting* if $\rho(n) \in F$, in which case we say that the word $w$ is *accepted* by $\mathcal{A}$. The language $L(\mathcal{A})$ of $\mathcal{A}$ is the set of words in $\Sigma^*$ accepted by $\mathcal{A}$.

**Transition systems.** We fix a set $\mathsf{ACT}$ of *action symbols*. A *transition system* over $\mathsf{ACT}$ is a tuple $\mathfrak{S} := \langle S; \{\rightarrow_a\}_{a \in \mathsf{ACT}} \rangle$, where $S$ is a set of *configurations* and $\rightarrow_a \subseteq S \times S$ is a binary relation over $S$. We use $\rightarrow$ to denote the relation $\bigcup_{a \in \mathsf{ACT}} \rightarrow_a$. We say that a sequence $s_1 \rightarrow \cdots \rightarrow s_{n+1}$ is a *path* in $\mathfrak{S}$ if $s_1, ..., s_{n+1} \in S$ and $s_i \rightarrow s_{i+1}$ for $i \in \{1, \ldots, n\}$. A transition system is called *bounded branching* if the number of configurations reachable from a configuration in one step is bounded. Formally, this means that there exists an *a priori* integer $N$ such that for all $s \in S$, $|\{s' \in S : s \rightarrow s'\}| \leq N$.

**Probabilistic transition systems.** A *probabilistic transition system* (*PTS*) [34] is a structure $\mathfrak{S} := \langle S; \{\delta_a\}_{a \in \mathsf{ACT}} \rangle$ where $S$ is a set of configurations and $\delta_a : S \to \mathcal{D}_S \cup \{\overline{0}\}$ maps each configuration to either a probability distribution or a zero function $\overline{0}$. Here $\delta_a(s) = \overline{0}$ simply means that $s$ is a "dead end" for action $a$. We shall use $\delta_a(s, s')$ to denote $\delta_a(s)(s')$. In this paper, we always assume that $\delta_a(s, s')$ is a rational number and $|\{s' : \delta_a(s, s') \neq 0\}| < \infty$. The *underlying transition graph* of a PTS is a transition system $\langle S; \{\rightarrow_a\}_{a \in \mathsf{ACT}} \rangle$ such that $s \rightarrow_a s'$ iff $\delta_a(s, s') \neq 0$.

It is standard (e.g. see [34]) to impose the *minimal probability assumption* on the PTS that we shall be dealing with, i.e., there is $\epsilon > 0$ such that any transition with a non-zero probability $p$ satisfies $p > \epsilon$. This assumption is practically sensible since it is satisfied by most PTSs that we deal with in practice (e.g. finite PTS, probabilistic pushdown automata [21], and most examples from probabilistic parameterized systems [36, 37] including our examples from Section 5). The minimal probability assumption,

among others, implies that the PTS is bounded-branching (i.e. that its underlying transition graph is bounded-branching). In the sequel, we shall adopt this assumption.

**Probabilistic bisimulations.** Let $\mathfrak{S} := \langle S; \{\delta_a\}_{a\in\mathsf{ACT}}\rangle$ be a PTS. We write $s \xrightarrow{\rho}_a S'$ if $\sum_{s'\in S'} \delta_a(s,s') = \rho$. A *probabilistic bisimulation* for $\mathfrak{S}$ is an equivalence relation $R$ over $S$, such that $(p,q) \in R$ implies

$$\forall a \in \mathsf{ACT}.\, \forall S' \in S/R.\, (p \xrightarrow{\rho}_a S' \Leftrightarrow q \xrightarrow{\rho}_a S'). \tag{1}$$

We say that $p$ and $q$ are *probabilistic bisimilar* (written as $p \sim q$) if there is a probabilistic bisimulation $R$ such that $(p,q) \in R$. We can compute probabilistic bisimulation between two PTSs $\mathfrak{S} := \langle S; \{\delta_a\}_{a\in\mathsf{ACT}}\rangle$ and $\mathfrak{S}' := \langle S'; \{\delta'_a\}_{a\in\mathsf{ACT}}\rangle$ by computing a probabilistic bisimulation $R$ for the disjoint union of $\mathfrak{S}$ and $\mathfrak{S}'$, which is defined as $\mathfrak{S} \sqcup \mathfrak{S}' := \langle S \sqcup S'; \{\delta''_a\}_{a\in\mathsf{ACT}}\rangle$ where $\delta''_a(s) := \delta_a(s)$ for $s \in S$, and $\delta''_a(s) := \delta'_a(s)$ for $s \in S'$. In such case, we say $R$ is a probabilistic bisimulation between $\mathfrak{S}$ and $\mathfrak{S}'$.

## 3 Framework of Regular Relations

In this section we describe the framework of regular relations for specifying probabilistic infinite-state systems, properties to verify, and proofs, all in a uniform symbolic way. The framework is amenable to automata-theoretic algorithms in the spirit of *regular model checking* [3, 13].

The framework of *regular relations* [8] (a.k.a. *automatic relations* [9]) uses the first-order theory of universal[1] automatic structure

$$\mathfrak{U} := \langle \Sigma^*; \preceq, \mathrm{eqL}, \{l_a\}_{a\in\Sigma}\rangle, \tag{2}$$

where $\Sigma$ is some finite alphabet, $\preceq$ is the (non-strict) prefix-of relation, eqL is the binary equal length predicate, and $l_a$ is a unary predicate asserting that the last letter of the word is $a$. The domain of the structure is the set of finite words over $\Sigma$, and for words $w, w' \in \Sigma^*$, we have $w \preceq w'$ iff there is some $w'' \in \Sigma^*$ such that $w \cdot w'' = w'$, $\mathrm{eqL}(w,w')$ iff $|w| = |w'|$, and $l_a(w)$ iff there is some $w'' \in \Sigma^*$ such that $w = w'' \cdot a$.

Next, we discuss the expressive power of first-order formulas over the universal automatic structures, and decision procedures for satisfiability of such formulas. In Section 4, we shall describe: (1) how to specify a PTS as a first-order formula in $\mathfrak{U}$, and (2) how to specify the verification condition for probabilistic bisimulation property in this theory. In Section 5, we shall show that the theory is sufficiently powerful for capturing probabilistic bisimulations for interesting examples.

*Expressiveness and Decidability.* The name "regular" associated with this framework is because the set of formulas $\varphi(x_1, \ldots, x_k)$ first-order definable in $\mathfrak{U}$ coincides with *regular relations*, i.e., relations definable by synchronous automata. More precisely, we define $[\![\varphi]\!]$ as the relation which contains all tuples $(w_1, \ldots, w_k) \in (\Sigma^*_\perp)^k$ such that $\mathfrak{U} \models \varphi(w_1, \ldots, w_k)$. In addition, we define the *convolution* $w_1 \otimes \cdots \otimes w_k$ of words

[1] Here, "universal" simply means that all automatic structures are first-order interpretable in this structure.

$w_1, \ldots, w_k \in \Sigma^*$ as a word $w$ over $\Sigma_\perp^k$ (where $\perp \notin \Sigma$) such that $w[i] = (a_1, \ldots, a_k)$ with

$$a_j = \begin{cases} w_j[i] & \text{if } |w_j| \geq i, \text{ or} \\ \perp & \text{otherwise.} \end{cases}$$

In other words, $w$ is obtained by juxtaposing $w_1, \ldots, w_k$ and padding the shorter words with $\perp$. For example, $010 \otimes 00 = (0,0)(1,0)(0,\perp)$. A $k$-ary relation $R$ over $\Sigma^*$ is *regular* if the set $\{w_1 \otimes \cdots \otimes w_k : (w_1, \ldots, w_k) \in R\}$ is a regular language over the alphabet $\Sigma_\perp^k$. The relationship between $\mathfrak{U}$ and regular relations can be formally stated as follows.

**Proposition 1 ([8–10]).**

1. *Given a formula $\varphi(\bar{x})$ over $\mathfrak{U}$, the relation $[\![\varphi]\!]$ is effectively regular. Conversely, given a regular relation $R$, we can compute a formula $\varphi(\bar{x})$ over $\mathfrak{U}$ such that $[\![\varphi]\!] = R$.*
2. *The first-order theory of $\mathfrak{U}$ is decidable.*

The decidability of the first-order theory of $\mathfrak{U}$ follows using a standard automata-theoretic algorithm (e.g. see [9, 49]).

In the sequel, we shall also use the term regular relations to denote relations definable in $\mathfrak{U}$. In addition, to avoid notational clutter, we shall freely use other regular relations (e.g. successor relation $\prec_{succ}$ of the prefix $\preceq$, and membership in a regular language) as syntactic sugar.

We note that the first-order theory of $\mathfrak{U}$ can also be reduced to weak monadic theory WS1S of one successor (therefore, allowing highly optimized tools like MONA [31] and Gaston [23]) by translating finite words to finite sets. The relationship between the universal automatic structure and WS1S can be made precise using the notion of *finite-set interpretations* [19, 46].

## 4 Probabilistic Bisimilarity within Regular Relations

In this section, we show how the framework of regular relations can be used to encode a PTS, and the corresponding proof rules for probabilistic bisimulation.

### 4.1 Specifying a probabilistic transition system

Since we assume that all probability values specified in our systems are rational numbers, the fact that our PTS is bounded-branching implies that we can specify the probability values by natural *weights* (by multiplying the probability values by the least common multiple of the denominators). For example, if a configuration $c$ has an action $toss$ that takes it to $c_1$ and $c_2$, each with probability $1/2$, then the new system simply changes both values of $1/2$ to 1. This is a known trick in the literature of probabilistic verification (e.g. see [1]). Therefore, we can now assume that the transition probability functions have range $\mathbb{N}$. *The challenge now is that our encoding of a PTS in the universal automatic structure must encode two different sorts* as words over a finite alphabet $\Sigma$: configurations and natural weights.

Now we are ready to show how to specify a PTS $\mathfrak{S}$ in our framework. Fix a finite alphabet $\Sigma$ containing at least two letters 0 and 1. We encode the domain of $\mathfrak{S}$ as words over $\Sigma$. In addition, a natural weight $n \in \mathbb{N}$ can be encoded in the usual way as a binary string. This motivates the following definition.

**Definition 1.** *Let $\mathfrak{S}$ be a PTS $\langle S; \{\delta_a\}_{a \in \mathsf{ACT}} \rangle$. We say that $\mathfrak{S}$ is regular if the domain $S$ is a regular subset of $\Sigma^*$ (i.e. definable by a first-order formula $\varphi(x)$ with one free variable over $\mathfrak{U}$), and if the graph of each function $\delta_a$ is a ternary regular relation (i.e. definable by a first-order formula $\varphi(x, y, z)$ over $\mathfrak{U}$, where $x$ and $y$ encode configurations, and $z$ encodes a natural weight).*

Definition 1 is quite general since it allows for an infinite number of different natural weights in the PTS. Note that we can make do without the second sort (of numeric weights) if we have only finitely many numeric weights $n_1, \ldots, n_m$. This can be achieved by specifying a regular relation $R_{a,i}$ for each action label $a \in \mathsf{ACT}$ and numeric weight $n_i$ with $i \in \{1, \ldots, m\}$.

*Example 1.* We show a regular encoding of a very simple PTS: a random walk on the set of natural numbers. At each position $x$, the system can non-deterministically choose to loop or to move. If the system chooses to loop, it will stay at the same position with probability 1. If the system chooses to move, it will move to $x+1$ with probability $1/4$, or move to $\max(0, x-1)$ with probability $3/4$. Normalising the probability values by multiplying by 4, we obtain the numeric weights of 4, 1, and 3 for the aforementioned transitions, respectively.

To represent the system by regular relations, we encode the positions in unary and the numeric weights in binary. The set of configurations is the regular language $1^*$. The graph of the transition probability function can be described by a first-order formula $\varphi(x, y, z) := \varphi_{\mathrm{loop}}(x, y, z) \vee \varphi_{\mathrm{move}}(x, y, z)$ over $\mathfrak{U}$, where

$$\begin{aligned}
\varphi_{\mathrm{loop}}(x, y, z) &:= x \in 1^* \wedge y \in 1^* \wedge ((x = y \wedge z = 100) \vee (x \neq y \wedge z = 0))\,; \\
\varphi_{\mathrm{move}}(x, y, z) &:= x \in 1^* \wedge y \in 1^* \wedge ((x \prec_{succ} y \wedge z = 1) \vee \\
&\qquad (y \prec_{succ} x \wedge z = 11) \vee (x = \varepsilon \wedge y = \varepsilon \wedge z = 11) \vee \\
&\qquad (\neg(x \prec_{succ} y) \wedge \neg(y \prec_{succ} x) \wedge \neg(x = \varepsilon \wedge y = \varepsilon) \wedge z = 0)).
\end{aligned}$$

□

*Example 2.* As a second example, consider a PTS (from [25], Example 1) described by a probabilistic pushdown automaton with states $Q = \{p, q, r\}$ and stack symbols $\Gamma = \{X, X', Y, Z\}$. There is a unique action $a$, and the transition rules $\delta_a$ are as follows:

$$\begin{array}{llll}
pX \xrightarrow{0.5} qXX & pX \xrightarrow{0.5} p & qX \xrightarrow{1} pXX & rY \xrightarrow{1} rXX \\
rX \xrightarrow{0.3} rYX & rX \xrightarrow{0.2} rYX' & rX \xrightarrow{0.5} r & \\
rX' \xrightarrow{0.4} rYX & rX' \xrightarrow{0.1} rYX' & rX' \xrightarrow{0.5} r &
\end{array}$$

A configuration of the PTS is a word in $Q\Gamma^*$, consisting of a state in $Q$ and a word over the stack symbols. A transition can be applied if the prefix of the configuration matches the left hand side of the transition rules above. We encode the PTS as follows: the set of configurations is $Q\Gamma^*$, the weights are represented in binary after normalization, and

the transition relation $\varphi(x, y, z)$ encodes the transition rules in disjunction. For example, the disjunct corresponding to the rule $pX \xrightarrow{0.5} qXX$ is

$$x \in Q\Gamma^* \wedge y \in Q\Gamma^* \wedge (\exists u.\, x = pXu \wedge y = qXXu) \wedge z = 101.$$

Note that the PTS is bounded branching with a bound 3. □

### 4.2 Proof rules for probabilistic bisimulation

Fix the set $\mathsf{ACT}$ of action symbols and the branching bound $N \geq 1$, owing to the minimal probability assumption. Consider a two-sorted vocabulary $\sigma = \langle \{P_a\}_{a \in \mathsf{ACT}}, R, + \rangle$, where $P_a$ is a ternary relation (with the first two arguments over the first sort, and the third argument over the second sort of natural numbers), $R$ is a binary relation over the first sort, and $+$ is the addition function over the second sort of natural numbers. The main result we shall show next is summarized in the following theorem:

**Theorem 1.** *There is a fixed first-order formula $\Phi$ over $\sigma$ such that a binary relation $R$ is a probabilistic bisimulation over a bounded-branching PTS $\mathfrak{S} = \langle S; \{\delta_a\}_{a \in \mathsf{ACT}} \rangle$ iff $(\mathfrak{S}, R) \models \Phi$. Furthermore, when $\mathfrak{S}$ is a regular PTS and $R$ is a regular relation, we can compute in polynomial time a first-order formula $\Phi'$ over $\mathfrak{U}$ such that $R$ is a probabilistic bisimulation over $\mathfrak{S}$ iff $\mathfrak{U} \models \Phi'$.*

This theorem implies the following result:

**Theorem 2.** *Given a regular relation $E \subseteq \Sigma^* \times \Sigma^*$ and a bounded-branching regular PTS $\mathfrak{S} = \langle S; \{\delta_a\}_{a \in \mathsf{ACT}} \rangle$, there exists an algorithm that either finds $(u, v) \in E$ which are not probabilistically bisimilar or finds a regular probabilistic bisimulation relation $R$ over $\mathfrak{S}$ such that $E \subseteq R$ if one exists. The algorithm does not terminate iff $E$ is contained in some probabilistic bisimulation relation but every probabilistic bisimulation $R$ containing $E$ is not regular.*

Note that when verifying parameterized systems we are typically interested in checking bisimilarity over *a set of pairs* (instead of just one pair) of configurations, and hence $E$ in the above statement.

*Proof (of Theorem 2).* To prove this, we provide two semi-algorithms, one for checking the existence of $R$ and the other for showing that a pair $(v, w) \in E$ is a witness for non-bisimilarity.

By Theorem 1, we can enumerate all possible candidate regular relation $R$ and effectively check that $R$ is a probabilistic bisimulation over $\mathfrak{S}$. The condition that $E \subseteq R$ is a first-order property, and so can be checked effectively.

To show non-bisimilarity is recursively enumerable, observe that if we fix $(v, w) \in E$ and a number $d$, then the restrictions $\mathfrak{S}_v$ and $\mathfrak{S}_w$ to configurations that are of distance at most $d$ away from $v$ and $w$ (respectively) are finite PTS. Therefore, we can devise a semi-algorithm which enumerates all $(v, w) \in E$, and all probabilistic modal logic (PML) formulas [34] $F$ over $\mathsf{ACT}$ containing only rational numbers (i.e. a formula of the form $\langle a \rangle_\mu F'$, where $\mu \in [0, 1]$ is a rational number, which is sufficient because we assume only rational numbers in the PTS). We need to check that $\mathfrak{S}_v, v \models F$, but $\mathfrak{S}_w, w \not\models F$. Model checking PML formulas over finite systems is decidable (in fact, the logic is subsumed by Probabilistic CTL [7]), which makes our check effective.

### 4.3 Proof of Theorem 1

In the rest of the section, we shall give a proof of Theorem 1. Given a binary relation $R \subseteq S \times S$, we can write a first-order formula $F_{eq}(R)$ for checking that $R$ is an equivalence relation:

$$\forall s, t, u \in S.\ R(s,s) \wedge (R(s,t) \Rightarrow R(t,s)) \wedge ((R(s,t) \wedge R(t,u) \Rightarrow R(s,u)).$$

We shall next define a formula $\varphi_a(p,q)$ for each $a \in \mathsf{ACT}$, such that $R$ is a probabilistic bisimulation for $\mathfrak{S} = \langle S; \{\delta_a\}_{a\in\mathsf{ACT}}\rangle$ iff $(\mathfrak{S}, R) \models \Phi(R)$, where

$$\Phi(R) := F_{eq}(R) \wedge \forall p, q \in S.\ R(p,q) \Rightarrow \bigwedge\nolimits_{a\in\mathsf{ACT}} (\psi_a(p) \wedge \psi_a(q)) \vee \varphi_a(p,q). \quad (3)$$

The formula $\psi_a(s) := \forall s' \in S.\ \delta_a(s,s') = 0$ states that configuration $s$ cannot move to any configuration through action $a$.

Before we describe $\varphi_a(p,q)$, we provide some intuition and define some intermediate macros. Fix configurations $p$ and $q$. Informally, $\varphi_a(p,q)$ will first guess a set of configurations $u_1, \ldots, u_N$ containing the successors of $p$ on action $a$, and a set of configurations $v_1, \ldots, v_N$ containing the successors of $q$ on action $a$. Second, it will guess labellings $\alpha_1, \ldots, \alpha_N$ and $\beta_1, \ldots, \beta_N$ which correspond to partitionings of the configurations $u_1, \ldots, u_N$ and $v_1, \ldots, v_N$, respectively. The intuition is that the $\alpha$'s and $\beta$'s "name" the partitions: if $\alpha_i = \alpha_j$ (resp. $\beta_i = \beta_j$), then $u_i$ and $u_j$ (resp. $v_i$ and $v_j$) are guessed to be in the same partition. The formula then checks that the guessed partitioning is compatible with the equivalence relation $R$ (i.e. if the labelling claims $u_i$ and $u_j$ are in the same partition, then indeed $R(u_i, u_j)$ holds), and that the probability masses of the partitions assigned by configurations $p$ and $q$ satisfy the constraint given in (1).

For the first part, we define a formula

$$\mathsf{succ}_a(w; u_1, \ldots, u_N) := \left(\bigwedge\nolimits_{1\le i<j\le N} u_i \neq u_j\right) \wedge \left(\forall u \in S.\ \delta_a(w,u) \neq 0 \Rightarrow \bigvee\nolimits_{1\le i\le N} u = u_i\right),$$

stating that the successors of configuration $w$ on action $a$ are among the $N$ distinct configurations $u_1, \ldots, u_N$. Note that a configuration may have fewer than $N$ successors. In this case, we can set the rest of the variables to arbitrary distinct configurations.

For the second part, we shall check that $R$ is compatible with the guessed partitions, and that configurations $p$ and $q$ assign the same probability mass to the same partition. Let $k_1, \ldots, k_n$ be a labelling for configurations $s_1, \ldots, s_n$. To check that the partitioning induced by the labelling is compatible with $R$, we need to express the condition that $k_i = k_j$ if and only if $R(s_i, s_j)$ holds. To this end, we define a formula

$$\mathsf{compat}_R(s_1, \ldots, s_n; k_1, \ldots, k_n) := \bigwedge\nolimits_{1\le i<j\le n} (R(s_i, s_j) \Leftrightarrow k_i = k_j)\,.$$

Now, we are ready to define $\varphi_a(p,q)$:

$$\begin{aligned}\varphi_a(p,q) := \exists u_1, \ldots, u_N, v_1, \ldots, v_N \in S.\ \exists \alpha_1, \ldots, \alpha_N, \beta_1, \ldots, \beta_N \in \mathbb{N}.\\ \mathsf{succ}_a(p; u_1, \ldots, u_N) \wedge \mathsf{succ}_a(q; v_1, \ldots, v_N) \wedge \\ \mathsf{compat}_R(u_1, \ldots, u_N, v_1, \ldots, v_N; \alpha_1, \ldots, \alpha_N, \beta_1, \ldots, \beta_N) \wedge \\ \forall k \in \mathbb{N}.\ \left(\sum\nolimits_{i:\ \alpha_i = k} \delta_a(p, u_i) = \sum\nolimits_{i:\ \beta_i = k} \delta_a(q, v_i)\right).\end{aligned} \quad (4)$$

With this definition, $\varphi_a(p,q)$ holds if and only if $p \xrightarrow{\rho}_a S' \Leftrightarrow q \xrightarrow{\rho}_a S'$ holds for any $\rho \geq 0$ and equivalence class $S' \in S/R$.

*Example 3.* Consider the PTS from Example 2. The configurations $pXZ$ and $rX$ are probabilistic bisimilar. This can be seen using a probabilistic bisimulation relation with equivalence classes $\{pX^kZ\} \cup \{rw : w \in \{X, X'\}^k\}$ for all $k \geq 0$ and $\{qX^{k+1}Z\} \cup \{rYw : w \in \{X, X'\}^k\}$ for all $k \geq 1$. The probabilistic bisimulation relation is definable as the symmetric closure of a regular relation $R$, where $(w_1, w_2) \in R$ iff

$$\begin{aligned}
&(w_1 = w_2) \vee \\
&(w_1 \in pX^*Z \wedge w_2 \in r(X + X')^*\bot \wedge |w_1| = |w_2|) \vee \\
&(w_1 \in r(X + X')^* \wedge w_2 \in r(X + X')^* \wedge |w_1| = |w_2|) \vee \\
&(w_1 \in qX^*Z \wedge w_2 \in rY(X + X')^*\bot \wedge |w_1| = |w_2|) \vee \\
&(w_1 \in rY(X + X')^* \wedge w_2 \in rY(X + X')^* \wedge |w_1| = |w_2|).
\end{aligned}$$

For this example, the formula (3) simplifies to $F_{eq}(R) \wedge \forall s, t \in S.\ \varphi_a(p,q)$ for the unique action $a$. This formula defines a condition that checks the bisimulation relation for all states symbolically. To see the formula in action, fix configurations $pXZ$ and $rX$ which are probabilistic bisimilar. In the PTS, $pXZ$ has two successors, $qXXZ$ and $pZ$, each with probability $0.5$, and $rX$ has three successors, $rYX$ with probability $0.3$, $rYX'$ with probability $0.2$, and $r$ with probability $0.5$. In the formula for $\varphi_a(p,q)$, we can set the successors $u_i$ of $pXZ$ and the successors $v_j$ of $rX$ as above (the third "successor" $u_3$ is set to an arbitrary configuration not reachable from $pXZ$), and set $\alpha_1 = 1$, $\alpha_2 = 2$, $\beta_1 = \beta_2 = 1$, and $\beta_3 = 2$, corresponding to the equivalence classes of the bisimulation relation. One can check that the probability masses to these classes are the same.

We remark that the first-order theory of $\mathfrak{U}$ is sufficient to encode any probabilistic pushdown automaton, not just this example. □

We proceed to show that if $R$ and $\delta_a$ are first-order definable over $\mathfrak{U}$ then so are $\psi_a$ and $\varphi_a$. Suppose that $\delta_a$ is encoded using the ternary relation $\delta_a(x, y, z)$, as stated in the previous section. (We shall re-use the symbol $\delta$ here to avoid a clash of names.)

We define $\psi_a(s) := \forall s' \in S.\ \forall z \in \mathbb{N}.\ \delta_a(s, s', z) \Leftrightarrow z = 0$. To define $\varphi_a$, the key is to express the sum of transition probabilities in the logic. We use the fact that addition of integers in binary encoding is regular (see e.g. [9]), and write a formula that performs iterated addition. Formally, for each $a \in \mathsf{ACT}$ we define a formula $\chi_a$ such that

$$\begin{aligned}
&\chi_a(u; u_1, \ldots, u_N; \alpha_1, \ldots, \alpha_N; k; z) := \\
&\qquad \exists z_1, \ldots, z_{N+1} \in \mathbb{N}.\ z_1 = 0 \wedge z_{N+1} = z \wedge \textstyle\bigwedge_{1 \leq i \leq N} \chi'_a(u, u_i, \alpha_i, k, z_i, z_{i+1}),
\end{aligned}$$

where

$$\chi'_a(u, u', \kappa, k, x, y) := (\kappa = k \wedge \exists z.\ \delta_a(u, u', z) \wedge y = x + z) \vee (\kappa \neq k \wedge y = x)$$

performs a single addition—we use the fact that addition "$y = x + z$" in binary is encodable as a regular relation—and $z_1, \ldots, z_{N+1}$ store the intermediate sums. Hence,

given $k \in \mathbb{N}$, $u_1, \ldots, u_N, v_1, \ldots, v_N \in S$, and $\alpha_1, \ldots, \alpha_N, \beta_1, \ldots, \beta_N \in \mathbb{N}$,

$$\sum\nolimits_{i:\ \alpha_i = k} \delta_a(p, u_i) = \sum\nolimits_{i:\ \beta_i = k} \delta_a(q, v_i)$$

if and only if

$$\exists z \in \mathbb{N}.\ \chi_a(p; u_1, \ldots, u_N; \alpha_1, \ldots, \alpha_N; k; z) \wedge \chi_a(q; v_1, \ldots, v_N; \beta_1, \ldots, \beta_N; k; z).$$

It follows that $\varphi_a(p, q)$ defined in (4) can be encoded in the first-order theory of $\mathfrak{U}$. This concludes our proof of Theorem 1.

*Remark.* Note that it is decidable to check whether a given presentation of a regular PTS is valid. To see this, suppose that a set $\Delta := \{\delta_a(x, y, z)\}_{a \in \mathsf{ACT}}$ of formulae is claimed to encode the probabilistic transition functions of a PTS with a branching bound $N$. Fix a formula $\delta_a \in \Delta$. First, we need to check that for all $x \in S$, there are at most $N$ distinct $y$'s such that $\delta_a(x, y, z)$ satisfies $z \neq 0$. Second, we need to check that $[\![\delta_a]\!]$ is a function, i.e., $\forall x, y.\ \exists! z.\ \delta_a(x, y, z)$, where $\exists! z.\ \varphi(\bar{x}, z)$ is a shorthand for the formula asserting there exists precisely one $z$ such that $\varphi(\bar{x}, z)$ is true. Third, we need to check that $[\![\delta_a]\!]$ encodes a mapping $S \to \{\overline{0}\} \cup \mathcal{D}_S$. The first two requirements are easily seen to be expressible as a first-order formula and hence is algorithmic over $\mathfrak{U}$. The third requirement amounts to checking the assertion that there exists $w_a \in \mathbb{N}$ satisfying

$$\begin{aligned}\forall x \in S.\ (\forall y \in S.\ \forall z \in \mathbb{N}.\ \delta_a(x, y, z) \Leftrightarrow z = 0) \vee \\ (\exists y_1, \ldots, y_N \in S.\ \exists z_1, \ldots, z_N \in \mathbb{N}. \\ \mathsf{succ}_a(x; y_1, \ldots, y_N) \wedge \bigwedge\nolimits_{1 \leq i \leq N} \delta_a(x, y_i, z_i) \wedge \sum\nolimits_{1 \leq i \leq N} z_i = w_a),\end{aligned}$$

which is a first-order formula and is algorithmic over $\mathfrak{U}$ by the fact that summation of a fixed number of weights is regular (as shown earlier in this section). Finally, since all of the $w_a$'s are expected to be the same common multiple of the denominators of the transition probabilities, we need to check that there is $w \in \mathbb{N}$ such that $w_a = w$ for all $a \in \mathsf{ACT}$. This is again algorithmic as we can pinpoint the exact value of each $w_a$ by enumeration.

## 5 Application to Anonymity Verification

In this section, we show how to verify the anonymity property of cryptographic protocols via computation of probabilistic bisimulations. We shall first formalize the connection between the concepts of anonymity and probabilistic bisimulation. We then introduce a verification framework and apply it to verify the anonymity property of the dining cryptographers protocol [16] and the grades protocol [29].

A *(discrete time) Markov chain* (a.k.a. *DTMC*) is a structure $\mathfrak{M} := \langle S; \delta; L \rangle$ where $S$ is a set of configurations, $\delta : S \to \mathcal{D}_S$ is a family of probability distributions, and $L : S \to \mathsf{ACT}$ is a labelling of the states. We shall use $\delta(s, s')$ to denote $\delta(s)(s')$, the transition probability from $s$ to $s'$. A sequence $s_0 \ldots s_n \in S^*$ is called a *path* of $\mathfrak{M}$ if $\delta(s_i, s_{i+1}) > 0$ for $i \in \{0, \ldots, n-1\}$. The probability distribution induced

by the paths in a DTMC can be defined using a standard cylinder construction (see e.g. [33]) as follows. Given a finite path $\pi := s_0 \cdots s_n \in S^*$, we set $Run_\pi$ to be a *basic cylinder*, which is the set of all finite/infinite paths with $\pi$ as a prefix. We associate this cylinder with probability $\Pr^{s_0}(Run_\pi) = \prod_{i=0}^{n-1} \delta(s_i, s_{i+1})$. This gives rise to a unique probability measure for the $\sigma$-algebra over the set of all paths from $s_0$.

Given a PTS $\mathfrak{S} := \langle S; \{\delta_a\}_{a \in \mathsf{ACT}} \rangle$, an *adversary* $f : S^* \to \mathsf{ACT}$ resolves the non-determinacy of $\mathfrak{S}$ and induces a DTMC $\mathfrak{S}_f := \langle S'; \delta'; L' \rangle$. Here $S' := S^* \cup \{\$\}$ contains all finite paths of $\mathfrak{S}$ plus a "sink state" $\$$ such that $\delta'(\pi) := I_\$$[2] if and only if either $\pi = \$$, or $\delta_{f(\pi)}$ is the zero function. We define $\delta'(\pi) := \delta_{f(\pi)}$ otherwise. The labelling of $\mathfrak{S}_f$ is defined as $L'(\$) := \bot$ and $L'(\pi) := f(\pi)$ for $\pi \in S^*$.

Given a DTMC $\langle S; \delta; L \rangle$, the *trace* of a path $\pi := s_0 \cdots s_n \in S^*$ is defined as $\tau(\pi) := L(s_0) \cdots L(s_n)$. A *trace event* $\mathcal{T}$ is a set of finite traces; the probability of $\mathcal{T}$ with respect to a configuration $s$ is specified with $\Pr^s(\mathcal{T}) := \Pr^s(\bigcup\{Run_\pi : \tau(\pi) \in \mathcal{T}, \pi \text{ starts from } s\})$.

Now we are ready to define the concept of anonymity. Fix $\mathfrak{S} := \langle S; \{\delta_a\}_{a \in \mathsf{ACT}} \rangle$ and a set $\mathcal{I} \subseteq S$ of initial configurations. We say $\mathfrak{S}$ is *anonymous to an adversary* $f$ if for all $s \in \mathcal{I}$ and trace event $\mathcal{T}$, the value of $\Pr^s(\mathcal{T})$ in $\mathfrak{S}_f$ is solely determined by $\mathcal{T}$. Intuitively, this means that the adversary cannot obtain any information about a specific initial configuration by experimenting on the system and observing the traces.

We shall only consider external adversaries in this paper. An adversary $f : S^* \to \mathsf{ACT}$ is *external* if $f(s_0 \cdots s_n) = f(s'_0 \cdots s'_n)$ when $L(s_i) = L(s'_i)$ for $i \in \{0, \ldots, n\}$. That is, an external adversary takes action solely based on the trace she has observed so far. We call a PTS *anonymous* if it is anonymous to any external adversary. The following result establishes a connection between the anonymity property and probabilistic bisimulations.

**Proposition 2.** *Let $\mathfrak{S} := \langle S; \{\delta_a\}_{a \in \mathsf{ACT}} \rangle$ be a PTS and $f$ be an external adversary for $\mathfrak{S}$. Then for all $u, v \in S$ such that $u \sim v$, $\Pr^u(\mathcal{T}) = \Pr^v(\mathcal{T})$ holds for any trace event $\mathcal{T}$ in $\mathfrak{S}_f$. That is, configurations $u$ and $v$ induce the same trace distribution in $\mathfrak{S}_f$.*

Based on Proposition 2, we propose a framework to verify the anonymity property of $\mathfrak{S}$ as follows. We first specify a "reference system" $\mathfrak{S}' := \langle S; \{\delta'_a\}_{a \in \mathsf{ACT}} \rangle$ that has the same initial configurations and actions as those of $\mathfrak{S}$, except that the trace distribution of $\mathfrak{S}'_f$ is independent of specific initial configurations for any adversary $f$. We then try to find a bisimulation relation $R$ between $\mathfrak{S}$ and the reference system $\mathfrak{S}'$ satisfying $R \supseteq \{(s, s') \in \mathcal{I} \times \mathcal{I}' : s = s'\}$. When such a relation $R$ is found, we can conclude that the trace distribution of $\mathfrak{S}_f$ is also independent of the initial configurations for any adversary $f$, and hence prove the anonymity property of $\mathfrak{S}$.

**The dining cryptographers protocol.** Dining cryptographers protocol [16] is a multi-party computation algorithm aiming to securely compute the XOR of the secret bits held by the participants. More precisely, consider a ring of $n \geq 3$ participants $p_0, \ldots, p_{n-1}$ such that each participant $p_i$ holds a secret bit $x_i$. To compute $x_0 \oplus \cdots \oplus x_{n-1}$ without revealing information about the values of $x_0, \ldots, x_{n-1}$, the participants carry out a

[2] Recall that $I_s$ denotes the point distribution at $s$, namely $I_s(s) = 1$.

two-stage computation as follows: i) Each two adjacent participants $p_i$, $p_{i+1}$ compute a random bit $b_i$ that is accessible only to them; ii) Each participant $p_i$ announces the value $a_i := x_i \oplus b_i \oplus b_{i-1}$[3] to the other participants. Hence, every participant $p_i$ can observe the values of $x_i$, $b_i$, $b_{i-1}$ and $a_0, \ldots, a_{n-1}$. It turns out that $a_0 \oplus \cdots \oplus a_{n-1} = x_0 \oplus \cdots \oplus x_{n-1}$, so all participants are able to compute the XOR of the secret bits after executing the protocol. Furthermore, the anonymity property of the protocol assures that any individual participant $p_i$ cannot infer the values of the other secret bits from the information she has observed during the execution of the protocol.

We model the protocol as a length-preserving regular PTS. The configurations of a ring of $n$ participants are encoded as words of size $n$. The initial configurations are words $w \in \{0,1\}^*$ such that $w[i]$ represents $x_i$ for $i \in \{0, \ldots, |w| - 1\}$. The transition relation consists of six transitions: observer non-deterministically tossing head (via action $\mathsf{head}$), observer non-deterministically tossing tail (via action $\mathsf{tail}$), non-observer tossing head with probability 0.5 (via action $\mathsf{toss}$), non-observer tossing tail with probability 0.5 (via action $\mathsf{toss}$), participant announcing zero (via action $\mathsf{zero}$), and participant announcing one (via action $\mathsf{one}$). The outcomes of the tosses by the observer are visible (i.e. as actions $\mathsf{head}$ and $\mathsf{tail}$), while the outcomes of the tosses by the other participants are hidden (i.e. as action $\mathsf{toss}$). Each maximal trace from an initial configuration of size $n$ consists of $n$ successive tossing actions, followed by $n$ successive announcing actions. Starting from an initial configuration $w$ and for $i \in \{0, \ldots, n-1\}$, the $i$-th toss action updates the value of $w[j]$ to $w[j] \oplus b_i$ for $j \in \{i, i+1\}$, where $b_i = 1$ if a head is tossed and $b_i = 0$ otherwise. Any configuration $v$ reached after $n$ tosses would satisfy $v[i] = x_i \oplus b_i \oplus b_{i-1}$ for $i \in \{0, \ldots, n-1\}$. The PTS then "prints out" the configuration by going through $n$ announcement transitions via actions $a_0, \ldots, a_{n-1}$, such that $a_i$ is $\mathsf{one}$ if $v[i] = 1$ and $a_i$ is $\mathsf{zero}$ if $v[i] = 0$.

We consider the case where the first participant of the protocol is the observer. The maximal traces of the PTS in this case are in form of $t \cdot t'$, where $|t| = |t'|$, $t \in \{\mathsf{head}, \mathsf{tail}\}\,\mathsf{toss}^*\{\mathsf{head}, \mathsf{tail}\}$, and $t' \in \{\mathsf{zero}, \mathsf{one}\}^*$. For example, $\mathsf{head}\ \mathsf{toss}\ \mathsf{tail}\ \mathsf{one}\ \mathsf{zero}\ \mathsf{zero}$ is a maximal trace starting from initial configuration $010$. To prove anonymity, we define a reference system such that the initial configurations and the actions are the same as those of the original PTS, except that the announcements $a_0, \ldots, a_{n-1}$ encoded in the maximal traces from an initial configuration $w$ are uniformly distributed over $\{(a_0, \ldots, a_{n-1}) : a_0 \oplus \cdots \oplus a_{n-1} = w[0] \oplus \cdots \oplus w[n-1],\ a_0 = w[0] \oplus b_0 \oplus b_{n-1}\}$.[4] In this way, the distribution of the announcements is independent of the initial configuration once the values of $x_0 \oplus \cdots \oplus x_{n-1}$, $x_0$, $b_0$, and $b_{n-1}$ (i.e. the information revealed to the first participant) are fixed. We then compute a probabilistic bisimulation between the original system and the reference system, establishing the anonymity property that the first participant cannot infer the secret bits of the other participants from the information she observes.

[3] All arithmetical operations on the subscripts are performed modulo $n$ to take the ring structure into account.

[4] Such a distribution can be obtained by i) choose $a_1, \ldots, a_{n-2} \in \{0,1\}$ uniformly at random; ii) set $a_0 = w[0] \oplus b_0 \oplus b_{n-1}$; iii) set $a_{n-1} = a_0 \oplus \cdots \oplus a_{n-2} \oplus w[0] \oplus \cdots \oplus w[n-1]$.

*A generalized dining cryptographers protocol.* We have also considered a generalized dining cryptographers protocol where the secret messages $x_0, \ldots, x_{n-1}$ of the $n$ participants are bit-vectors of the same size. Note that the set of the initial configurations is not regular when the size of the secret messages is parameterized. To construct a regular model, we allow a configuration to encode secret messages of different sizes, and devise the transition system such that an initial configuration $w$ can finish the protocol (i.e. can have a trace containing all of the announcements $a_0, \ldots, a_{n-1}$) if and only if the messages encoded in $w$ have same size. The resulting PTS is a regular system; it over-approximates the PTS of the generalized dining cryptographers protocol in the sense that the anonymity property of the former implies that of the latter.

**The grades protocol.** The grades protocol [29] is a multi-party computation algorithm aiming to securely compute the sum of the secrets held by the participants. The setting of the protocol is pretty similar to that of the dining cryptographers: given $n \geq 3$ and $g \geq 2$, we have a ring of $n$ participants $p_0, \ldots, p_{n-1}$ where each participant $p_i$ holds a secret $x_i \in \{0, \ldots, g-1\}$. Note that both $g$ and $n$ are parameterized in this protocol. The goal of the participants is to compute the sum $x_0 + \cdots + x_{n-1}$ without revealing information about the individual secrets. Define $M := (g-1) \cdot n + 1$. The protocol consists of two steps: i) Each two adjacent participants $p_i$, $p_{i+1}$ compute a random number $y_i \in \{0, \ldots, M-1\}$; ii) Each participant $p_i$ announces $a_i := (x_i + y_i - y_{i-1}) \bmod M$ to the other participants. After executing the protocol, the participants compute $a := a_0 + \cdots + a_{n-1} \bmod M$. Because of the ring structure, the $y_i$'s will be cancelled out in the sum. Thus the value of $a$ will equal to the sum of all secrets. The anonymity property of the protocol asserts that no participant can infer the secrets held by the other participants from the information she has observed.

We consider a variant of the grades protocol where $M$ can be any power of two greater than $(g-1) \cdot n$. Observe that the same anonymity and correctness property of the original protocol also holds for this variant. To verify the anonymity property, we model an over-approximation of the protocol where the secrets are allowed to range over $\{0, \ldots, M-1\}$. This model is similar to the one we have constructed for the generalized dining cryptographers protocol except that, e.g., the XOR operations are now replaced with bitwise additions and negations. A reference system is specified such that the announcements $a_1, \ldots, a_{n-1}$ observed by the first participant $p_0$ are uniformly distributed over the values satisfying $a_0 + \cdots + a_{n-1} \bmod M = x_0 + \cdots + x_{n-1} \bmod M$. By computing a probabilistic bisimulation between the original system and the reference system, we establish the anonymity property that the grades protocol is anonymous whenever $M$ is chosen as a power of two with $M \geq (g-1) \cdot n + 1$.

## 6 Learning Probabilistic Bisimulations

We propose an automata learning method to automatically compute regular probabilistic bisimulations $R$, focusing on the case of *length-preserving* PTSs, which covers all examples given in the previous section. The approach uses active automata learning, for instance Angluin's $L^*$ method [5] or refinements of it, to compute $R$.

**Algorithm 1:** Equivalence check for $L^*$

**Input:** Candidate automaton $\mathcal{H}$ over $\Sigma \times \Sigma$, PTS $\mathfrak{S}$, and regular relation $E \subseteq (\Sigma \times \Sigma)^*$.

**Result:** $NoSolution(v, w)$ if there is no bisimulation $R$ with $E \subseteq R$;
$PositiveCEX(v, w)$ if $\mathcal{H}$ should accept $(v, w)$, but does not;
$NegativeCEX(v, w)$ if $\mathcal{H}$ accepts $(v, w)$, but should not;
$Correct$ if $\mathcal{H}$ encodes a correct bisimulation for $\mathfrak{S}$ and $E \subseteq \mathcal{L}(\mathcal{H})$.

Check whether $E \subseteq \mathcal{L}(\mathcal{H})$, and whether $\mathfrak{S} \models \Phi(\mathcal{L}(\mathcal{H}))$ using the $\Phi$ from (3);
**if** *there is a counterexample of minimal length* $n$ **then**
  Compute the greatest bisimulation $\bar{R}_n$ restricted to configurations of length $n$;
  **if** *there is* $(v, w) \in E \setminus \bar{R}_n$ *with* $|v| = |w| = n$ **then**
    Output $NoSolution(v, w)$ and abort;
  **else if** *there is* $(v, w) \in \mathcal{L}(\mathcal{H}) \setminus \bar{R}_n$ *with* $|v| = |w| = n$ **then**
    **return** $NegativeCEX(v, w)$;
  **else if** *there is* $(v, w) \in \bar{R}_n \setminus \mathcal{L}(\mathcal{H})$ **then**
    **return** $PositiveCEX(v, w)$;
**else**
  **return** $Correct$;

This approach is inspired by previous work on using active automata learning for invariant inference [18, 54]. Our procedure assumes (i) as input a bounded-branching PTS $\mathfrak{S} = \langle S; \{\delta_a\}_{a \in \mathsf{ACT}} \rangle$, as well as a length-preserving regular relation $E \subseteq (\Sigma \times \Sigma)^*$ supposed to be covered by $R$; (ii) an effective way to check the correctness of $R$, i.e., a decision procedure in the sense of Theorem 1; and (iii) a procedure to compute the greatest probabilistic bisimulation $\bar{R}_n \subseteq (\Sigma \times \Sigma)^n$ for $\mathfrak{S}$ restricted to configurations of any length $n \in \mathbb{N}$. The last assumption can easily be satisfied for length-preserving PTSs. Indeed, such systems, restricted to configurations of length $n$, are finite-state, so that efficient existing methods [6, 17, 20, 52] apply. A solution $R$ is presented as a deterministic letter-to-letter transducer, i.e., as a deterministic finite-state automaton over the alphabet $\Sigma \times \Sigma$.

Since $L^*$-style learning requires the taught language to be uniquely defined, our approach attempts to learn a representation of the greatest *length-preserving* probabilistic bisimulation relation $\bar{R} \subseteq (\Sigma \times \Sigma)^*$, which is the unique bisimulation relation formed by the union of all length-preserving probabilistic bisimulations of $\mathfrak{S}$, i.e., $\bar{R} = \bigcup_{n \geq 1} \bar{R}_n$. Because $\bar{R}$ is not in general computable, the learning process might diverge and fail to produce any probabilistic bisimulation. It can also happen that learning terminates, but yields a probabilistic bisimulation relation strictly smaller than $\bar{R}$.

The $L^*$ method requires a teacher that is able to answer two kinds of queries:

- **membership queries**, i.e., whether a pair $(v, w)$ of words should be accepted by the automaton to be learned. Since our learner tries to learn the greatest bisimulation, the teacher can answer this query by checking whether the configurations $v, w$ are bisimilar; this is done by computing the greatest bisimulation $\bar{R}_{|v|}$ restricted to configurations of any length $|v| = |w|$, and checking whether or not $(v, w) \in \bar{R}_{|v|}$.
- **equivalence queries**, i.e., whether a candidate automaton $\mathcal{H}$ is the correct language to be learned. Such queries can essentially be answered by checking whether the

language $\mathcal{L}(\mathcal{H})$ satisfies the formula $\Phi(R)$ from (3). The complete algorithm for answering equivalence queries is given in Algorithm 1. The algorithm first attempts to find a shortest counterexample to the proof rule. If a counterexample of length $n$ is found, then the difference set $\mathcal{L}(\mathcal{H}) \mathbin{\Delta} \bar{R}_n$ must contain at least one pair of length $n$. Any of such pairs is a valid counterexample for automata learning since the learner tries to learn the greatest bisimulation. The teacher thus reports one such pair to be a positive or negative counterexample according to its membership in $\bar{R}_n$.

*Properties of the Learning Algorithm.* The learning procedure terminates when the teacher outputs *NoSolution* or returns *Correct* for an equivalence query. In the former case, the teacher explicitly provides a pair of non-bisimilar configurations in $E$. In the latter case, the procedure computes an automaton $\mathcal{H}$ such that $E \subseteq \mathcal{L}(\mathcal{H})$ and $\mathcal{L}(\mathcal{H})$ is a correct probabilistic bisimulation (as it satisfies the proof rule based on Theorem 1), though not necessarily the greatest one. Since all counterexamples reported by the teacher are contained in $\mathcal{L}(\mathcal{H}) \mathbin{\Delta} \bar{R}$, the learning procedure is guaranteed to terminate for PTSs where the greatest probabilistic bisimulation $\bar{R}$ is regular.

*Optimization with Inductive Invariants.* There is a natural way to optimize the learning procedure by only considering a *regular* inductive invariant $Inv$ such that $Inv$ contains the set of reachable configurations and $E \subseteq Inv \times Inv$. The optimization is done by simply replacing the greatest finite-length bisimulations $\bar{R}_i$ in Algorithm 1, and when answering membership queries, with the greatest bisimulation $\bar{R}_i^I = \bar{R}_i \cap Inv$ on the inductive invariant. Since $\bar{R}_i^I$ can be a lot smaller than $\bar{R}_i$, this can lead to significant speed-ups. Note that a bisimulation $R'$ on $Inv$ can be extended to a bisimulation $R$ on all configurations by setting $R = R' \cup \{(v, v) : v \notin Inv\}$. The inductive invariant $Inv$ may be manually specified, or automatically generated using techniques like in [18, 54].

*Experimental Results and Conclusion.* We have implemented a prototype in Scala to test our learning method. Given a PTS specified over $\mathfrak{U}$, our tool first translates it to WS1S formulas and obtains finite automata for these formulas using the Mona tool [30]. Our prototype then applies the $L^*$ learning procedure as described in this section, including the optimization to consider only the configurations of valid format. When answering an equivalence query, our tool invokes Mona to verify candidate automata and obtain counterexamples (line 1-2 of Algorithm 1). We use the prototype tool to prove the anonymity property of the three protocols described in Section 5. The proofs generated by our tool are finite-state automata encoding the desired probabilistic bisimulation relations. The experimental results are summarized in Table 1.

## References


1. P. A. Abdulla, N. B. Henda, and R. Mayr. Decisive markov chains. *Logical Methods in Computer Science*, 3(4), 2007.
2. P. A. Abdulla, B. Jonsson, M. Nilsson, J. d'Orso, and M. Saksena. Regular model checking for LTL(MSO). *STTT*, 14(2):223–241, 2012.
3. P. A. Abdulla, B. Jonsson, M. Nilsson, and M. Saksena. A survey of regular model checking. In *CONCUR*, pages 35–48, 2004.

| Case study | #states | #trans | mona | bisim | total |
|---|---|---|---|---|---|
| Dining Cryptographers, single-bit | 13 | 832 | 2s | 2s | 6s |
| Dining Cryptographers, multi-bit | 16 | 1024 | 3s | 24s | 28s |
| The grades protocol | 25 | 1600 | 5s | 28s | 35s |

**Table 1.** Experimental results. For each case study, we list the size of the final proof produced by our tool, the time taken by Mona to verify the candidate automata, the time taken by our tool to compute the fixed-length bisimulations, and the total computation time of the learning procedure. Experiments are run on a Windows laptop with 2.4GHz Intel i5 processor and 2GB memory limit.


4. W. Ahrendt, B. Beckert, R. Bubel, R. Hähnle, P. H. Schmitt, and M. Ulbrich, editors. *Deductive Software Verification - The KeY Book - From Theory to Practice*, volume 10001 of *Lecture Notes in Computer Science*. Springer, 2016.
5. D. Angluin. Learning regular sets from queries and counterexamples. *Inf. Comput.*, 75(2):87–106, Nov. 1987.
6. C. Baier. Polynomial time algorithms for testing probabilistic bisimulation and simulation. In *Computer Aided Verification, 8th International Conference, CAV '96, New Brunswick, NJ, USA, July 31 - August 3, 1996, Proceedings*, pages 50–61, 1996.
7. C. Baier and J.-P. Katoen. *Principles of Model Checking*. MIT Press, 2008.
8. M. Benedikt, L. Libkin, T. Schwentick, and L. Segoufin. Definable relations and first-order query languages over strings. *J. ACM*, 50(5):694–751, 2003.
9. A. Blumensath. Automatic Structures. Diploma thesis, RWTH-Aachen, 1999.
10. A. Blumensath and E. Grädel. Finite presentations of infinite structures: Automata and interpretations. *Theory Comput. Syst.*, 37(6):641–674, 2004.
11. B. Boigelot, A. Legay, and P. Wolper. Iterating transducers in the large (extended abstract). In *CAV*, pages 223–235, 2003.
12. F. Bonchi and D. Pous. Checking NFA equivalence with bisimulations up to congruence. In *The 40th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL '13, Rome, Italy - January 23 - 25, 2013*, pages 457–468, 2013.
13. A. Bouajjani, B. Jonsson, M. Nilsson, and T. Touili. Regular model checking. In *Computer Aided Verification, 12th International Conference, CAV 2000, Chicago, IL, USA, July 15-19, 2000, Proceedings*, pages 403–418, 2000.
14. A. R. Bradley and Z. Manna. *The Calculus of Computation: Decision Procedures with Applications to Verification*. Springer, 1998.
15. K. Chatzikokolakis, G. Norman, and D. Parker. Bisimulation for demonic schedulers. In *Foundations of Software Science and Computational Structures, 12th International Conference, FOSSACS 2009, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2009, York, UK, March 22-29, 2009. Proceedings*, pages 318–332, 2009.
16. D. Chaum. The dining cryptographers problem: Unconditional sender and recipient untraceability. *Journal of cryptology*, 1(1):65–75, 1988.
17. D. Chen, F. van Breugel, and J. Worrell. On the complexity of computing probabilistic bisimilarity. In *Foundations of Software Science and Computational Structures - 15th International Conference, FOSSACS 2012, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2012, Tallinn, Estonia, March 24 - April 1, 2012. Proceedings*, pages 437–451, 2012.
18. Y. Chen, C. Hong, A. W. Lin, and P. Rümmer. Learning to prove safety over parameterised concurrent systems. In *2017 Formal Methods in Computer Aided Design, FMCAD 2017, Vienna, Austria, October 2-6, 2017*, pages 76–83, 2017.

19. T. Colcombet and C. Löding. Transforming structures by set interpretations. *Logical Methods in Computer Science*, 3(2), 2007.
20. S. Derisavi, H. Hermanns, and W. H. Sanders. Optimal state-space lumping in markov chains. *Inf. Process. Lett.*, 87(6):309–315, 2003.
21. J. Esparza and K. Etessami. Verifying probabilistic procedural programs. In *FSTTCS 2004: Foundations of Software Technology and Theoretical Computer Science, 24th International Conference, Chennai, India, December 16-18, 2004, Proceedings*, pages 16–31, 2004.
22. R. Fagin, J. Y. Halpern, Y. Moses, and M. Y. Vardi. *Reasoning About Knowledge*. MIT Press, Cambridge, MA, USA, 2003.
23. T. Fiedor, L. Holík, P. Janků, O. Lengál, and T. Vojnar. Lazy automata techniques for WS1S. In *TACAS'17*, volume 10205 of *LNCS*, pages 407–425. Springer, 2017.
24. C. Flanagan, K. Leino, M. Lillibridge, G. Nelson, J. B. Saxe, and R. Stata. Extended static checking for Java. In *PLDI 02: Programming Language Design and Implementation*, pages 234–245. ACM, 2002.
25. V. Forejt, P. Jancar, S. Kiefer, and J. Worrell. Bisimilarity of probabilistic pushdown automata. In *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2012, December 15-17, 2012, Hyderabad, India*, pages 448–460, 2012.
26. P. Garg, D. Neider, P. Madhusudan, and D. Roth. Learning invariants using decision trees and implication counterexamples. In *Proceedings of the 43rd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2016, St. Petersburg, FL, USA, January 20 - 22, 2016*, pages 499–512, 2016.
27. P. Habermehl and T. Vojnar. Regular model checking using inference of regular languages. *Electr. Notes Theor. Comput. Sci.*, 138(3):21–36, 2005.
28. A. Hinton, M. Z. Kwiatkowska, G. Norman, and D. Parker. PRISM: A tool for automatic verification of probabilistic systems. In *Tools and Algorithms for the Construction and Analysis of Systems, 12th International Conference, TACAS 2006 Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2006, Vienna, Austria, March 25 - April 2, 2006, Proceedings*, pages 441–444, 2006.
29. S. Kiefer, A. S. Murawski, J. Ouaknine, B. Wachter, and J. Worrell. APEX: an analyzer for open probabilistic programs. In *Computer Aided Verification - 24th International Conference, CAV 2012, Berkeley, CA, USA, July 7-13, 2012 Proceedings*, pages 693–698, 2012.
30. N. Klarlund and A. Møller. *Mona version 1.4: User manual*. BRICS, Department of Computer Science, University of Aarhus Denmark, 2001.
31. N. Klarlund, A. Møller, and M. I. Schwartzbach. MONA implementation secrets. *International Journal of Foundations of Computer Science*, 13(4):571–586, 2002. World Scientific Publishing Company. Earlier version in Proc. 5th International Conference on Implementation and Application of Automata (CIAA) 2000, Springer-Verlag LNCS vol. 2088.
32. S. Kundu, Z. Tatlock, and S. Lerner. Proving optimizations correct using parameterized program equivalence. In *Proceedings of the 30th ACM SIGPLAN Conference on Programming Language Design and Implementation*, PLDI '09, pages 327–337, New York, NY, USA, 2009. ACM.
33. M. Z. Kwiatkowska. Model checking for probability and time: from theory to practice. In *18th IEEE Symposium on Logic in Computer Science (LICS 2003), 22-25 June 2003, Ottawa, Canada, Proceedings*, page 351, 2003.
34. K. G. Larsen and A. Skou. Bisimulation through probabilistic testing. *Inf. Comput.*, 94(1):1–28, 1991.
35. K. R. M. Leino. Dafny: An automatic program verifier for functional correctness. In E. M. Clarke and A. Voronkov, editors, *Logic for Programming, Artificial Intelligence, and Reasoning - 16th International Conference, LPAR-16, Dakar, Senegal, April 25-May 1, 2010,*

*Revised Selected Papers*, volume 6355 of *Lecture Notes in Computer Science*, pages 348–370. Springer, 2010.
36. O. Lengál, A. W. Lin, R. Majumdar, and P. Rümmer. Fair termination for parameterized probabilistic concurrent systems. In *TACAS*, pages 499–517, 2017.
37. A. W. Lin and P. Rümmer. Liveness of randomised parameterised systems under arbitrary schedulers. In *CAV'16 (2)*, volume 9779 of *LNCS*, pages 112–133. Springer, 2016.
38. C. Löding, P. Madhusudan, and D. Neider. Abstract learning frameworks for synthesis. In *Tools and Algorithms for the Construction and Analysis of Systems - 22nd International Conference, TACAS 2016, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2016, Eindhoven, The Netherlands, April 2-8, 2016, Proceedings*, pages 167–185, 2016.
39. R. Milner. *Communication and Concurrency*. Prentice Hall, 1989.
40. D. Neider and N. Jansen. Regular model checking using solver technologies and automata learning. In *NASA Formal Methods, 5th International Symposium, NFM 2013, Moffett Field, CA, USA, May 14-16, 2013. Proceedings*, pages 16–31, 2013.
41. D. Neider and U. Topcu. An automaton learning approach to solving safety games over infinite graphs. In *TACAS*, pages 204–221, 2016.
42. M. Nilsson. *Regular Model Checking*. PhD thesis, Uppsala Universitet, 2005.
43. O. Padon, G. Losa, M. Sagiv, and S. Shoham. Paxos made EPR: decidable reasoning about distributed protocols. *PACMPL*, 1(OOPSLA):108:1–108:31, 2017.
44. O. Padon, K. L. McMillan, A. Panda, M. Sagiv, and S. Shoham. Ivy: safety verification by interactive generalization. In *Proceedings of the 37th ACM SIGPLAN Conference on Programming Language Design and Implementation, PLDI 2016, Santa Barbara, CA, USA, June 13-17, 2016*, pages 614–630, 2016.
45. PRISM case study: Dining Cryptographers. `http://www.prismmodelchecker.org/casestudies/diningcrypt.php`.
46. S. Rubin. *Automatic Structures*. PhD thesis, University of Auckland, New Zealand, 2004.
47. G. Sénizergues. The bisimulation problem for equational graphs of finite out-degree. *SIAM J. Comput.*, 34(5):1025–1106, 2005.
48. J. Srba. *Roadmap of Infinite results*, volume Vol 2: Formal Models and Semantics. World Scientific Publishing Co., 2004.
49. A. W. To. *Model Checking Infinite-State Systems: Generic and Specific Approaches*. PhD thesis, LFCS, School of Informatics, University of Edinburgh, 2010.
50. A. W. To and L. Libkin. Recurrent reachability analysis in regular model checking. In *LPAR*, pages 198–213, 2008.
51. A. W. To and L. Libkin. Algorithmic metatheorems for decidable LTL model checking over infinite systems. In *Foundations of Software Science and Computational Structures, 13th International Conference, FOSSACS 2010, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2010, Paphos, Cyprus, March 20-28, 2010. Proceedings*, pages 221–236, 2010.
52. A. Valmari and G. Franceschinis. Simple $O(m \log n)$ time markov chain lumping. In *Tools and Algorithms for the Construction and Analysis of Systems, 16th International Conference, TACAS 2010, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2010, Paphos, Cyprus, March 20-28, 2010. Proceedings*, pages 38–52, 2010.
53. A. Vardhan. *Learning To Verify Systems*. PhD thesis, Department of Computer Science, University of Illinois at Urbana-Champaign, 2006.
54. A. Vardhan, K. Sen, M. Viswanathan, and G. Agha. Learning to verify safety properties. In J. Davies, W. Schulte, and M. Barnett, editors, *Formal Methods and Software Engineering, 6th International Conference on Formal Engineering Methods, ICFEM 2004, Seattle,*

*WA, USA, November 8-12, 2004, Proceedings*, volume 3308 of *Lecture Notes in Computer Science*, pages 274–289. Springer, 2004.