Process Symmetry in Probabilistic Transducers

Shaull Almagor
Computer Science Department, Technion, Israel
shaull@cs.technion.ac.il

Abstract

Model checking is the process of deciding whether a system satisfies a given specification. Often, when the setting comprises multiple processes, the specifications are over sets of input and output signals that correspond to individual processes. Then, many of the properties one wishes to specify are symmetric with respect to the processes identities. In this work, we consider the problem of deciding whether the given system exhibits symmetry with respect to the processes' identities. When the system is symmetric, this gives insight into the behaviour of the system, as well as allows the designer to use only representative specifications, instead of iterating over all possible process identities.

Specifically, we consider probabilistic systems, and we propose several variants of symmetry. We start with precise symmetry, in which, given a permutation π, the system maintains the exact distribution of permuted outputs, given a permuted inputs. We proceed to study approximate versions of symmetry, including symmetry induced by small L_∞ norm, variants of Parikh-image based symmetry, and qualitative symmetry. For each type of symmetry, we consider the problem of deciding whether a given system exhibits this type of symmetry.

2012 ACM Subject Classification Theory of computation → Verification by model checking; Theory of computation → Abstraction; Theory of computation → Concurrency

Keywords and phrases Symmetry, Probabilistic Transducers, Model Checking, Permutations

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2020.43

Funding *Shaull Almagor*: Supported by a European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 837327.

Acknowledgements The author thanks Gal Vardi for discussions on the motivation for this work.

1 Introduction

A fundamental approach to automatic verification is *model checking* [4], where we are given a system and a specification, and we check whether all possible behaviours of the system satisfy the specification. In model checking of *reactive* systems, the specification is over sets of inputs I and outputs O, and the system is an I/O transducer, which takes sequences of inputs in 2^I, and responds with an output in 2^O. Then, model checking amounts to deciding whether for every input sequence, the matching output sequence generated by the transducer, satisfies the specification.

In practice, and especially in verification of concurrent systems, the input and output sets have some correspondence. For example, in an arbiter for k processes, the inputs are typically $I = \{i_1, \ldots, i_k\}$, where i_j is interpreted as "a request was generated by Process j", and the outputs are $O = \{o_1, \ldots, o_k\}$, where o_j is interpreted as "Process j was granted access". In such cases, specification often end up having symmetric repetitions of a similar pattern. For example, we may wish to specify that in our arbiter, if Process j_1 generated a request before Process j_2, then a grant for j_1 should be given before a grant for j_2. However, in order to specify this in e.g., LTL (Linear Temporal Logic), we would have to explicitly write this statement for every pair of processes j_1, j_2. In the worst case, this could entail a blowup of $k!$ in the size of the formula, which incurs a further exponential blowup during model-checking algorithms.

This drawback, however, vanishes when we consider a *symmetric* system: intuitively, a system is symmetric if permuting the input signals generates an output sequence of similarly permuted outputs. If a system satisfies this property, then it is enough to check whether it satisfies a representative specification. Indeed, any permutation of the processes is guaranteed to be equivalently satisfied.

Unfortunately, deterministic systems are unlikely to be completely symmetric, unless they are very naive (e.g., no grants are ever given). Indeed, tie-breaking in deterministic systems has an inherent asymmetry to it. In *probabilistic* systems, however, no asymmetry is needed to break ties – one can randomly choose a result.

In this paper, we consider several notions of symmetry for probabilistic transducers, and their corresponding decision procedures. We start with the most restrictive version of symmetry, in which a transducer $\mathcal{T}$ is symmetric under a permutation if the distribution of outputs that are generated for an input sequence x is identical to the distribution of permuted outputs for the permuted input sequence (Section 3). We show that deciding whether a transducer is symmetric under a given permutation is decidable in polynomial time, and use basic results in group theory to give a similar result for deciding whether a transducer is symmetric under all permutations in a permutation group.

We then proceed to study approximate notions of symmetry, in order to capture cases where a system is not fully symmetric, but still may exhibit some symmetrical properties. On the negative side, using results on probabilistic automata, we show that an L_∞ approximation variant of symmetry results in undecidability. On the positive side, we study two variants of symmetry that only take into account the Parikh image of the output signals, and we are able to use results on probabilistic automata with rewards to obtain efficient decidability of symmetry for these variants (Section 4).

Finally, we study a qualitative version of symmetry, which offers a coarse "nondeterministic" approximation of symmetry (Section 5). We show that deciding whether a system is qualitatively symmetric is PSPACE complete.

The notion of symmetry is not only appealing for symmetry reductions in specification, but also as a standalone feature for the *explainability* of model checking: standard model-checking algorithms can output a counterexample whenever a system does not satisfy its specification. This gives the designer insight as to what is wrong with either the system or the specification. On the other hand, when the result of model checking is that a system does satisfy its specification, no additional information is typically given. While this is "good news", a designer often wants some information as to "why" the system is correct. In particular, the designer may be concerned that the specifications were too easy to satisfy (e.g., in vacuous specifications [1]). In this case, symmetry provides some information. Indeed, symmetry can be easily witnessed (as we show in Remark 4), so the designer can be convinced that any weakness of the specification, or any flaw of the system, is not biased toward a specific process, and will arise regardless of a specific order of processes. In addition, it shows that if the system satisfies e.g., liveness properties, then it satisfies them with the same "good event intervals" regardless of process identities.

Related work

Process symmetry [3, 8, 6, 12] and more general symmetry reductions [16, 17, 19] have been studied since the 90's, typically in the context of alleviating the state-explosion problem. Symmetry can either be specified by the designer or user [13,24,25], or detected automatically [15,16,32].

A close approach to our work here is [12], where the problem of detecting process

symmetries is studied. There, however, parametrized deterministic systems are studied, which shift the focus to the pattern of given symmetries (rather than our fixed-length permutations), and does not concern probabilities.

Symmetry in the probabilistic setting was studied in [11, 5], where model checking of probabilistic systems exploits known symmetries to avoid a state blowup by considering a quotient of the system under the symmetry.

We remark that the works above typically focus on exact symmetries, and use them to reduce the state space, whereas the focus of this paper is to decide whether a symmetry exists, for various types of (not necessarily exact) symmetries, and to use the symmetry to avoid blowup in the specification, as well as to give the user insight regarding the correctness of the system.

Due to lack of space, some proofs appear in the appendix.

2 Preliminaries

Probabilities and Distributions

Consider a finite set S. A *distribution* over S is a function $\mu : S \to [0,1]$ such that $\sum_{s\in S} \mu(s) = 1$. We denote the space of all distributions over S by $\Delta(S)$. Given a distribution μ, an *event* is a subset[1] $E \subseteq S$, and its *probability* under μ is $\Pr(E) = \sum_{e\in E} \mu(e)$. For an element $s \in S$, the *Dirac distribution* $\mathbb{1}[s]$ is given by $\mathbb{1}[s](r) = \begin{cases} 1 & r = s, \\ 0 & r \neq s. \end{cases}$ The *support* of a distribution μ is $\mathsf{Supp}(\mu) = \{s \in S : \mu(s) > 0\}$.

Given sets $S_1, \ldots, S_n$ and distributions $\mu_1, \ldots, \mu_n$ such that $\mu_i \in \Delta_i$ for every $1 \leq i \leq n$, a natural *product distribution* μ is induced on the product space $S_1 \times \cdots \times S_n$ where $\mu(s_1, \ldots, s_n) = \prod_{i=1}^{n} \mu_i(s_i)$.

Probabilistic Transducers and Automata

Consider two finite sets I and O of input and output signals, respectively. An I/O *probabilistic transducer* (henceforth just *transducer*) is $\mathcal{T} = \langle I, O, S, s_0, \delta, \boldsymbol{\ell}\rangle$ where S is a finite set of states, s_0 is an initial state, $\delta : S \times 2^I \to \Delta(S)$ is a transition function, assigning to each (state,letter) pair a distribution of successor states, and $\boldsymbol{\ell} : S \to 2^O$ is a labelling function.

For a word $x = \mathbf{i}_1 \cdot \mathbf{i}_2 \cdots \mathbf{i}_n \in (2^I)^+$, a *run* of $\mathcal{T}$ on x is a sequence $\rho = q_0, q_1, \ldots, q_n$ where $q_0 = s_0$, and the *probability* of the run ρ is $\prod_{j=0}^{n-1} \delta(q_j, \mathbf{i}_{j+1})(q_{j+1})$. Note that indeed this induces a probability measure μ on $\{s_0\} \times S^n$ via the product distribution.

A run ρ is *proper* if $\rho \in \mathsf{Supp}(\mu)$. That is, if it has positive probability. We denote the space of proper runs by $\mathsf{runs}(\mathcal{T}, x)$. In the following, we usually refer only to proper runs, and we omit the term "proper" when it is clear from context. We extend the labelling function $\boldsymbol{\ell}$ to runs by $\boldsymbol{\ell}(\rho) = \boldsymbol{\ell}(q_1) \cdot \boldsymbol{\ell}(q_2) \cdots \boldsymbol{\ell}(q_n)$. Observe that we ignore the labelling of the initial state, and only consider nonempty words, to avoid edge cases.

For $x \in (2^I)^+$ and $y \in (2^O)^+$ such that $|x| = |y|$, we denote by $\mathcal{T}(x) = y$ the event $\{\rho \in \mathsf{runs}(\mathcal{T}, x) : \boldsymbol{\ell}(\rho) = y\}$. Thus, $\Pr(\mathcal{T}(x) = y)$ is the probability that the output generated by $\mathcal{T}$ on input x is exactly y. We denote by $x \otimes y \in (2^{I\cup O})^\omega$ the combined word $(\mathbf{i}_1 \cup \mathbf{o}_1) \cdot (\mathbf{i}_2 \cup \mathbf{o}_2) \cdots (\mathbf{i}_n \cup \mathbf{o}_n)$.

[1] In general E needs to be a *measurable subset*, but since we only consider finite sets, any subset is measurable.

The sets I and O are called *corresponding signals* if $I = \{i_1, \ldots, i_k\}$ and $O = \{o_1, \ldots, o_k\}$. Intuitively, for $1 \leq j \leq k$ we think of i_j as a request generated by a process j, and of o_j as a corresponding grant generated by the system.

A *probabilistic automaton (PA)* is $\mathcal{A} = \langle Q, \Sigma, \delta, q_0, F \rangle$ where Q is a finite set of states, Σ is a finite alphabet, $\delta : Q \times \Sigma \to \Delta(Q)$ is a probabilistic transition function, $q_0 \in Q$ is an initial state, and $F \subseteq Q$ is a set of accepting states. Similarly to transducers, an input word $x \in \Sigma^*$ induces a probability measure on the set $\mathsf{runs}(\mathcal{A}, x)$ of runs of $\mathcal{A}$ on x. Then, we denote by $\mathcal{A}(x)$ the probability that a run of $\mathcal{A}$ on x is accepted, i.e. ends in a state in F.

Permutations

We assume familiarity with basic notions in group theory (see e.g. [2]). A *permutation* of the set $[k] = \{1, \ldots, k\}$ is a bijection $\pi : [k] \to [k]$. A standard representation of permutations is by a *cycle decomposition*, where, for example, the cycle (1 2 7) represents the permutation π where $\pi(1) = 2, \pi(2) = 7, \pi(7) = 1$, and for all other elements we have $\pi(j) = j$. The set of all permutations on $[k]$, equipped with the functional composition operator $\circ$ forms the *symmetric group* $\mathcal{S}_k$. Any subgroup of $\mathcal{S}_k$ is referred to as a *permutation group*. A *generating set* of a permutation group G is a finite set $X = \{\pi_1, \ldots, \pi_m\}$ such that every permutation $\tau \in G$ can be expressed as a composition of the elements in X. For such a set X, we denote the group generated by it by $\langle X \rangle$. It is well known that $\{(1\ 2), (1\ 2\ \ldots\ k)\}$ is a generating set of $\mathcal{S}_k$ (see e.g., [2]).

Consider corresponding signals $I = \{i_1, \ldots, i_k\}$ and $O = \{o_1, \ldots, o_k\}$, and let $\pi \in \mathcal{S}_k$. For a letter $\mathbf{i} = \{i_{j_1}, \ldots, i_{j_m}\} \in 2^I$, we define $\pi(\mathbf{i}) = \{i_{\pi j_1, \ldots, i_{\pi(j_m)}}\}$. That is, π permutes the signals given in $\mathbf{i}$.[2] Then, for a word $x = \mathbf{i}_1 \cdot \mathbf{i}_2 \cdots \mathbf{i}_n \in (2^I)^+$, we define $\pi(x) = \pi(\mathbf{i}_1) \cdot \pi(\mathbf{i}_2) \cdots \pi(\mathbf{i}_n)$. Similar definitions hold for O. Unless explicitly stated otherwise, we henceforth assume I and O are corresponding signals.

3 Symmetric Probabilistic Transducers

Let $\mathcal{T} = \langle I, O, S, s_0, \delta, \boldsymbol{\ell} \rangle$ be an I/O transducer over $I = \{i_1, \ldots, i_k\}$ and $O = \{o_1, \ldots, o_k\}$, and let $\pi \in \mathcal{S}_k$. We say that $\mathcal{T}$ is π-symmetric if for every $x \in (2^I)^+$ and $y \in (2^O)^+$ it holds that $\Pr(\mathcal{T}(x) = y) = \Pr(\mathcal{T}(\pi(x)) = \pi(y))$. That is, $\mathcal{T}$ is π-symmetric if whenever we permute the input by π, the resulting distribution on outputs is permuted by π as well.

▶ **Example 1.** Consider a Round-Robin arbiter over three processes, as depicted in Figure 1. At each state, the arbiter looks for a request from a single processor j, and grants it if it is on, then moves to a state corresponding to process $j + 1 \pmod 3$. Observe that this is a deterministic transducer, except that the initial state is unspecified.

Consider the case where we let the state marked 001 be initial, which corresponds to letting the first process start. In this case, the transducer is not π-symmetric for $\pi = (1\ 2\ 3)$. Indeed, the input word 100 will generate output 100, but its permutation $\pi(100) = 010$ generates output $000 \neq \pi(100)$.

However, if we introduce a probabilistic initial state, that chooses each state of $100, 010, 001$ as the next state, each with probability $\frac{1}{3}$, the transducer becomes π-symmetric for any $\pi \in \mathcal{S}_3$. ◀

[2] Formally, we would actually need I to be an ordered set. However, the order will be implied by the naming convention, so we let I be a set.

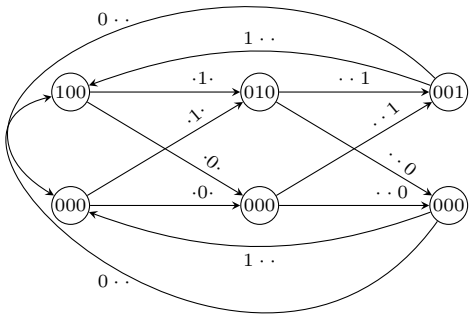


Figure 1 A transducer for a Round Robin arbiter. The labels on the transitions and states are the characteristic vectors of the labels, with · as placeholders. Thus, e.g., 100 is $\{i_1\}$, and $\cdot\cdot 1$ is any $\mathbf{i}$ such that $i_3 \in \mathbf{i}$. The initial state is unspecified, see Example 1.

Consider a permutation group $G = \langle X \rangle$ generated by $X = \{\pi_1, \ldots, \pi_m\}$. We say that $\mathcal{T}$ is G-symmetric if it is π-symmetric for every $\pi \in G$. Toward understanding symmetry, we start by showing that it is enough to consider symmetry under the generators.

▶ **Lemma 2.** *Consider an I/O transducer $\mathcal{T}$ over $I = \{i_1, \ldots, i_k\}$ and $O = \{o_1, \ldots, o_k\}$. If $\mathcal{T}$ is π-symmetric and τ-symmetric for $\pi, \tau \in \mathcal{S}_k$, then $\mathcal{T}$ is $\pi \circ \tau$-symmetric.*

Proof. Consider $x \in (2^I)^+$ and $y \in (2^I)^+$, we wish to show that $\Pr(\mathcal{T}(x) = y) = \Pr(\mathcal{T}(\pi(\tau(x))) = \pi(\tau(y)))$. Since $\mathcal{T}$ is τ-symmetric, then $\Pr(\mathcal{T}(x) = y) = \Pr(\mathcal{T}(\tau(x)) = \tau(y))$. Next, since $\mathcal{T}$ is π-symmetric, then applying the definition for the input $\tau(x) \in (2^I)^+$ and $\tau(y) \in (2^O)^+$, we have that $\Pr(\mathcal{T}(\tau(x)) = \tau(y)) = \Pr(\mathcal{T}(\pi(\tau(x))) = \pi(\tau(y)))$, and so overall $\Pr(\mathcal{T}(x) = y) = \Pr(\mathcal{T}(\pi(\tau(x))) = \pi(\tau(y)))$ and we are done. ◀

An immediate corollary of Lemma 2 is that in order to check whether $\mathcal{T}$ is G-symmetric, it suffices to check whether it is symmetric with respect to the generators of G.

▶ **Corollary 3.** *Consider an I/O transducer $\mathcal{T}$ and a permutation group G with generators X, then $\mathcal{T}$ is G-symmetric iff it is π-symmetric for every $\pi \in X$.*

▶ Remark 4 (Symmetry for Explainability). Corollary 3 is key to using symmetry for explainability of model checking. Indeed, it shows that we can convince a designer that a system is e.g., S_k-symmetric by showing that it is symmetric under the two generators. That is, the witness for symmetry consists of demonstrating symmetry on two permutations. As discussed in Section 1, once the designer is convinced the system possesses symmetric properties, she gains some insight to the possible reasons that make the system correct, or to possible behaviour of bugs, when the system is incorrect. ◀

The fundamental problem about symmetry of probabilistic transducers is whether a transducer is π-symmetric for a given permutation π. We now show that this problem can be solved in polynomial time.

▶ **Theorem 5.** *The problem of deciding, given an I/O transducer $\mathcal{T}$ and a permutation $\pi \in \mathcal{S}_k$, whether $\mathcal{T}$ is π-symmetric, is solvable in polynomial time.*

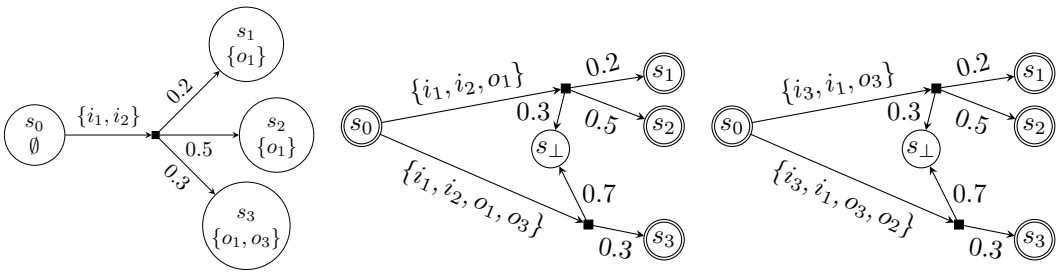
Proof. Given two probabilistic automata $\mathcal{A}$ and $\mathcal{B}$ over the alphabet Σ, the problem of determining whether $\mathcal{A}(x) = \mathcal{B}(x)$ for every $x \in \Sigma^*$, dubbed the *equivalence problem*, is solvable in polynomial time [7, 15, 18]. Our proof is by reduction of the problem at hand to the equivalence problem for probabilistic automata.

Consider an I/O transducer $\mathcal{T} = \langle I, O, S, s_0, \delta, \boldsymbol{\ell}\rangle$ over $I = \{i_1, \ldots, i_k\}$ and $O = \{o_1, \ldots, o_k\}$, and let $\pi \in \mathcal{S}_k$. We construct from $\mathcal{T}$ two PAs $\mathcal{A}$ and $\mathcal{B}$. Intuitively, $\mathcal{A}$ mimics the behaviour of $\mathcal{T}$, by reading words over $2^{I \cup O}$, and accepting a word $w \in (2^{I\cup O})^+$ with probability μ iff $\mathcal{T}$, when reading the inputs that appear in w, generates the outputs that appear in w with probability μ. The PA $\mathcal{B}$ works exactly like $\mathcal{A}$, but permutes both the inputs and outputs by π.

Formally, $\mathcal{A} = \langle S \cup \{q_\perp\}, 2^{I\cup O}, \eta, s_0, S\rangle$ and $\mathcal{B} = \langle S \cup \{q_\perp\}, 2^{I\cup O}, \zeta, s_0, S\rangle$ where $q_\perp$ is a new state, and the transition functions are defined as follows. Let $q \in S$ and $\sigma = \mathbf{i} \cup \mathbf{o}$ with $\mathbf{i} \in 2^I$ and $\mathbf{o} \in 2^O$, and let $V_p = \sum_{p\in S,\ \boldsymbol{\ell}(p)=\mathbf{o}} \delta(q, \mathbf{i})(p)$ be the probability assigned by $\mathcal{T}$ to seeing a state labelled $\mathbf{o}$ after reading $\mathbf{i}$ in state q, then $\eta(q, \sigma) \in \Delta(S \cup \{q_\perp\})$ is the following distribution:

$$\eta(q,\sigma)(p) = \begin{cases} \delta(q,\mathbf{i})(p) & \text{if } p \in S \text{ and } \boldsymbol{\ell}(p) = \mathbf{o} \\ 0 & \text{if } p \in S \text{ and } \boldsymbol{\ell}(p) \neq \mathbf{o} \\ 1 - V_p & \text{if } p = q_\perp \end{cases}$$

In addition, $\eta(q_\perp, \sigma)(q_\perp) = 1$ (so $q_\perp$ is a rejecting sink). We demonstrate the construction of $\mathcal{A}$ in Figures 2a and 2b.



(a) Transition in $\mathcal{T}$ **(b)** Transition in $\mathcal{A}$ **(c)** Transition in $\mathcal{B}$

Figure 2 A transition in a transducer $\mathcal{T}$ over $I = \{i_1, i_2, i_3\}$ and $O = \{o_1, o_2, o_3\}$, and the corresponding transitions in $\mathcal{A}$ and $\mathcal{B}$, under the permutation $\pi = (1\ 2\ 3)$. Observe that the transition in $\mathcal{B}$ corresponds to the inverse permutation, $\pi^{-1} = (3\ 2\ 1)$, so that e.g., $\pi(\{i_3, i_1\}) = \{i_1, i_2\}$.

The construction of $\mathcal{B}$ is similar, but accounts for the permutation π. Let $q \in S$ and $\sigma = \mathbf{i} \cup \mathbf{o}$ with $\mathbf{i} \in 2^I$ and $\mathbf{o} \in 2^O$, and let $U_p = \sum_{p\in S,\ \boldsymbol{\ell}(p)=\pi(\mathbf{o})} \delta(q, \pi(\mathbf{i}))(p)$ be the probability assigned by $\mathcal{T}$ to seeing a state labelled $\pi(\mathbf{o})$ after reading $\pi(\mathbf{i})$ in state q, then $\zeta(q, \sigma) \in \Delta(S \cup \{q_\perp\})$ is the following distribution:

$$\zeta(q,\sigma)(p) = \begin{cases} \delta(q,\pi(\mathbf{i}))(p) & \text{if } p \in S \text{ and } \boldsymbol{\ell}(p) = \pi(\mathbf{o}) \\ 0 & \text{if } p \in S \text{ and } \boldsymbol{\ell}(p) \neq \pi(\mathbf{o}) \\ 1 - U_p & \text{if } p = q_\perp \end{cases}$$

In addition, $\zeta(q_\perp, \sigma)(q_\perp) = 1$ (so $q_\perp$ is a rejecting sink). We demonstrate the construction of $\mathcal{B}$ in Figures 2a and 2c.

Consider words $x \in (2^I)^+$ and $y \in (2^O)^+$. Since $q_\perp$ is the only rejecting state in both $\mathcal{A}$ and $\mathcal{B}$, then by construction it is easy to see that $\mathcal{A}(x \otimes y) = \Pr(\mathcal{T}(x) = y)$ and $\mathcal{B}(x \otimes y) = \Pr(\mathcal{T}(\pi(x)) = \pi(y))$. Thus, we have that $\mathcal{A}$ and $\mathcal{B}$ are equivalent iff $\mathcal{T}$ is π-symmetric, and since equivalence can be decided in polynomial time, we are done. ◀

Combining Theorem 5 with Corollary 3, we have the following.

▶ **Corollary 6.** *The problem of deciding, given an I/O transducer $\mathcal{T}$ and a finite set of generators $X = \{\pi_1, \ldots, \pi_m\}$, whether $\mathcal{T}$ is $\langle X \rangle$-symmetric, is solvable in polynomial time.*

In particular, since the symmetric group $\mathcal{S}_k$ is generated by two permutations $\{(1\,2), (1\,2\,\ldots\,k)\}$, we have the following.

▶ **Corollary 7.** *The problem of deciding, given an I/O transducer $\mathcal{T}$, whether $\mathcal{T}$ is $\mathcal{S}_k$-symmetric, is solvable in polynomial time.*

4 Approximate Symmetry

While aspiring to obtain symmetric systems is noble, in practice exact symmetry may be too strong a requirement, for example if the source of randomness supplies binary bits, and one needs e.g., $\frac{1}{3}$ probability, then only an approximate probability can be used. Thus, it is reasonable to seek approximate notions of symmetry.

4.1 L_∞ Symmetry

The most straightforward approach toward approximate symmetry in probabilistic transducers is induced by the the L_∞ norm, as follows. Let $\mathcal{T}$ be an I/O-transducer, let $\pi \in \mathcal{S}_k$, and let $\epsilon > 0$. We say that $\mathcal{T}$ is (ϵ, π)-symmetric if $|\Pr(\mathcal{T}(x) = y) - \Pr(\mathcal{T}(\pi(x)) = \pi(y))| \leq \epsilon$ for every $x \in (2^I)^+$ and for every $y \in (2^O)^+$. That is, permuting the inputs by π perturbs the output distribution by at most ϵ.

Unfortunately, as we now show, approximate symmetry is undecidable.

▶ **Theorem 8.** *The problem of deciding, given an I/O transducer $\mathcal{T}$ a permutation $\pi \in \mathcal{S}_k$ and $\epsilon > 0$, whether $\mathcal{T}$ is (ϵ, π)-symmetric, is undecidable.*

Proof. The *emptiness problem* for PA is to decide, given a PA $\mathcal{A}$ over Σ and a threshold $\lambda \in [0, 1]$, whether there exists a word $w \in \Sigma^*$ such that $\mathcal{A}(w) > \lambda$. This problem is known to be undecidable [14, 13, 7].

We show that approximate symmetry is undecidable via a reduction from a restriction of the emptiness problem (or rather the complement thereof), where the given PA is over the alphabet $\{0, 1\}$. The problem remains undecidable under this restriction, as we can encode any larger alphabet Γ using fixed-length sequences in $\{0, 1\}^d$, such that while reading the d symbols that compose a single letter in Γ, the states are not accepting (and hence we do not introduce a word whose acceptance probability is above λ).

We start with an intuitive description of the reduction, depicted in Figure 3.

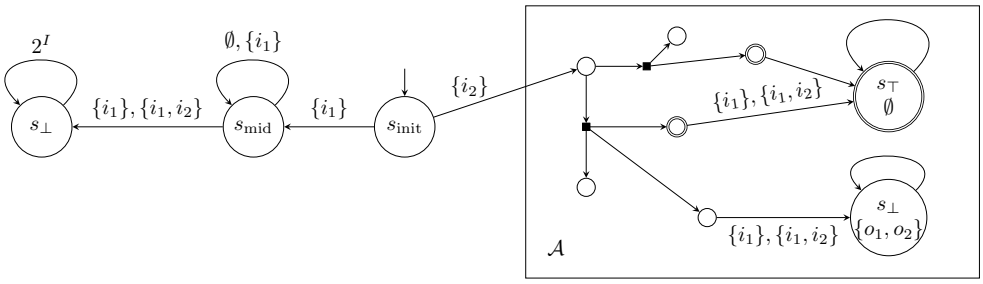


Figure 3 The transducer constructed from a PA. The black squares denote probabilistic branching.

Consider a PA $\mathcal{A}$ over the alphabet $\Sigma = \{0, 1\}$. We construct a transducer $\mathcal{T}$ over $I = \{i_1, i_2\}$ and $O = \{o_1, o_2\}$ which has two components. Initially, if $\mathcal{T}$ sees the input $\{i_2\}$,

it moves to a component which mimics $\mathcal{A}$ using the alphabet $\{\emptyset, \{i_2\}\}$ instead of $\{0,1\}$. At this stage, all the states are marked with the output $\{o_1, o_2\}$. If at any point the input signal i_1 is given, i.e. the letter $\{i_1\}$ or $\{i_1, i_2\}$, then $\mathcal{T}$ proceeds to a state labelled $\{o_1, o_2\}$ from non-accepting states of $\mathcal{A}$, and to a state labelled $\emptyset$ from accepting states. Thus, a word of the form $\{i_2\} \cdot x \cdot \{\{i_1\}, \{i_1, i_2\}\}^*$ with $x \in \{\emptyset, \{i_2\}\}^n$ would yield an output of the form $\emptyset^{n+1} \cdot \emptyset^*$ with probability $\mathcal{A}(x)$ and of the form $\emptyset^{n+1} \cdot \{o_1, o_2\}^*$ with probability $1 - \mathcal{A}(x)$. Observe that both output possibilities are invariant under the permutation (1 2).

If, initially, $\mathcal{T}$ sees the input $\{i_1\}$, it moves to a state labelled $\emptyset$, which loops as long as $\{i_1\}$ or $\emptyset$ are seen. Then, if $\{i_2\}$ or $\{i_1, i_2\}$ is seen, it moves to a sink labelled $\{o_1, o_2\}$. Essentially, this component mimics the output sequence of a rejecting run of $\mathcal{A}$ in the first component, under the permutation (1 2). Hence, taking $\epsilon = \lambda$, we have that $\mathcal{T}$ is $(\epsilon, (1\ 2))$-symmetric iff there does not exist a word x such that $\mathcal{A}(x) > \lambda$.

We proceed to give the precise reduction. Consider a PA $\mathcal{A} = \langle Q, \Sigma, \delta, q_0, F\rangle$ with $\Sigma = \{0,1\}$, we construct an I/O transducer $\mathcal{T} = \langle I, O, S, s_{\text{init}}, \eta, \boldsymbol{\ell}\rangle$ as follows. The states of $\mathcal{T}$ are $S = Q \cup \{s_{\text{mid}}, s_{\text{init}}, s_\top, s_\bot\}$, where $s_\bot \notin Q$, and the input and output sets are $I = \{i_1, i_2\}$ and $O = \{o_1, o_2\}$. The labelling function is given by $\boldsymbol{\ell}(q) = \emptyset$ for all $q \in Q$, $\boldsymbol{\ell}(s_\bot) = O = \{o_1, o_2\}$, and $\boldsymbol{\ell}(s_{\text{init}}) = \boldsymbol{\ell}(s_{\text{mid}}) = \{\emptyset\}$. The transition function, as depicted in Figure 3, is defined as follows.

First, for every $q \in Q$ and $\mathbf{i} \in \{\emptyset, \{i_2\}\}$, we have $\eta(q, \mathbf{i}) = \delta(q, \mathbf{i})$, where we identify $\{\emptyset, \{i_2\}\}$ with $\{0,1\}$ in an arbitrary bijective manner. Next, if $q \in F$, then $\eta(q, \{i_1\}) = \eta(q, \{i_1, i_2\}) = \mathbb{1}[s_\top]$, and if $q \notin F$ then $\eta(q, \{i_1\}) = \eta(q, \{i_1, i_2\}) = \mathbb{1}[s_\bot]$. The remaining transitions are

$$\eta(s_{\text{init}}, \{i_1\}) = \mathbb{1}[s_{\text{mid}}], \qquad \eta(s_{\text{mid}}, \emptyset) = \eta(s_{\text{mid}}, \{i_1\}) = \mathbb{1}[s_{\text{mid}}],$$
$$\eta(s_{\text{init}}, \{i_2\}) = \mathbb{1}[q_0], \qquad \eta(s_{\text{mid}}, \{i_2\}) = \eta(s_{\text{mid}}, \{i_1, i_2\}) = \mathbb{1}[s_\bot],$$
$$\eta(s_{\text{init}}, \emptyset) = \eta(s_{\text{init}}, \{i_1, i_2\}) = \mathbb{1}[s_\bot],$$

and for every $\mathbf{i} \in 2^I$ we have $\eta(s_\bot, \mathbf{i}) = \mathbb{1}[s_\bot]$ and $\eta(s_\top, \mathbf{i}) = \mathbb{1}[s_\top]$.

Let $\pi = (1\ 2)$ and $\epsilon = \lambda$. Keeping our identification of $\{\emptyset, \{i_2\}\}$ with $\{0,1\}$, we claim that there exists a word $x' \in \{\emptyset, \{i_2\}\}^*$ such that $\mathcal{A}(x') > \lambda$ iff there exists words $x \in (2^I)^+$ and $y \in (2^O)^+$ such that $|\Pr(\mathcal{T}(x) = y) - \Pr(\mathcal{T}(\pi(x)) = \pi(y))| > \epsilon$ (i.e. $\mathcal{T}$ is not (ϵ, π)-symmetric). Observe that $\boldsymbol{\ell}$ assigns only the labels $\emptyset$ and $\{o_1, o_2\}$, both of which are invariant under π. Thus, the latter condition becomes

$$|\Pr(\mathcal{T}(x) = y) - \Pr(\mathcal{T}(\pi(x)) = y)| > \epsilon. \tag{1}$$

We now turn to prove correctness. For the first direction, let $x' \in \{\emptyset, \{i_2\}\}^*$ such that $\mathcal{A}(x') > \lambda$, and consider the word $x = \{i_2\} \cdot x' \cdot \{i_1, i_2\}$. By the construction of $\mathcal{T}$, after seeing $\{i_2\}$, there is only a single run of $\mathcal{T}$ which proceeds to q_0. From there, $\mathcal{T}$ mimics the behaviour of $\mathcal{A}$ on x'. Thus, after reading x', the distribution of states has probability $\mathcal{A}(x)$ for states in F, and probability $1 - \mathcal{A}(x)$ in states in $Q \setminus F$. Note that up until then, only the label $\emptyset$ is seen, so the distribution of outputs is $\mathbb{1}[\emptyset^{|x'|+1}]$. Then, after reading $\{i_1, i_2\}$, the distribution of outputs give probability $\mathcal{A}(x)$ to $\emptyset^{|x'|+2}$, and $1 - \mathcal{A}(x)$ to $\emptyset^{|x'|+1} \cdot \{o_1, o_2\}$.

Now consider $\pi(x) = \{i_1\} \cdot \pi(x') \cdot \{i_1, i_2\}$. Upon reading $\{i_1\}$, the single run of $\mathcal{T}$ arrives at s_{mid}. Then, since $x' \in \{\emptyset, \{i_2\}\}^*$, we have that $\pi(x') \in \{\emptyset, \{i_1\}\}^*$, so the run of $\mathcal{T}$ stays in s_{mid}. Finally, reading $\{i_1, i_2\}$, the run moves to $s_\bot$. Therefore $\mathcal{T}(x)$ gives probability 1 to the output $\emptyset^{|x'|+1}\{o_1, o_2\}$. Thus, for the output $y = \emptyset^{|x'|+2}$, we have that $|\Pr(\mathcal{T}(x) = y) - \Pr(\mathcal{T}(\pi(x)) = y)| = |\mathcal{A}(x) - 0| > \lambda = \epsilon$, so $\mathcal{T}$ is not (ϵ, π)-symmetric.

For the converse direction, assume x, y are such that $|\Pr(\mathcal{T}(x) = y) - \Pr(\mathcal{T}(\pi(x)) = y)| > \epsilon$. We start by eliminating candidates for such x and y. First, observe that if x starts with $\emptyset$ or $\{ß_1, \varnothing_1\}$ (both of which are invariant under π), we have $\mathcal{T}(x)$ gives probability 1 to the output

$\ell(q_\perp)^{|x|} = \{o_1, o_2\}^{|x|}$, and so $\mathcal{T}(x) = \mathcal{T}(\pi(x))$, hence $|\Pr(\mathcal{T}(x) = y) - \Pr(\mathcal{T}(\pi(x)) = y)| = 0$ for all y, so this case cannot occur.

Next, we claim that without loss of generality, we can assume x starts with $\{i_2\}$. Indeed, if x starts with $\{i_1\}$, then $\pi(x)$ starts with $\{i_2\}$. Since $\pi(\pi(x)) = x$, we could start the argument with $\pi(x)$, while maintaining Equation (1).

Now, if x is of the form $\{i_2\} \cdot \{\emptyset, \{i_2\}\}^n$, then $\mathcal{T}(x)$ gives probability 1 to the output $\emptyset^{n+1}$, but $\pi(x)$ is now of the form $\{i_1\} \cdot \{\emptyset, \{i_1\}\}^n$, which also induces the same distribution, this case cannot occur as well.

It follows that x is of the form $\{i_2\} \cdot x' \cdot \{\{i_1\}, \{i_1, i_2\}\} \cdot (2^I)^*$ where $x' \in \{\emptyset, \{i_2\}\}^n$. We claim that $\mathcal{A}(x') > \lambda$. Indeed, as we observed above, $\mathcal{T}(x)$ gives probability $\mathcal{A}(x')$ to the output $\emptyset^{|x|}$ and probability $1 - \mathcal{A}(x')$ to the output $\emptyset^{|x'|+1} \cdot \{o_1, o_2\}^{|x|-|x'|-1}$. However, $\mathcal{T}(\pi(x))$ gives probability 1 to the output $\emptyset^{|x'|+1} \cdot \{o_1, o_2\}^{|x|-|x'|-1}$. Thus, there are only two possibilities for y in order for Equation (1) to hold: if $y = \emptyset^{|x|}$, we have

$$\lambda = \epsilon < |\Pr(\mathcal{T}(x) = y) - \Pr(\mathcal{T}(\pi(x)) = y)| = |\mathcal{A}(x') - 0| = \mathcal{A}(x')$$

and if $y = \emptyset^{|x'|+1} \cdot \{o_1, o_2\}^{|x|-|x'|-1}$, then

$$\lambda = \epsilon < |\Pr(\mathcal{T}(x) = y) - \Pr(\mathcal{T}(\pi(x)) = y)| = |1 - \mathcal{A}(x') - 1| = \mathcal{A}(x')$$

So in either case $\mathcal{A}(x') > \lambda$, and we are done. ◀

A-priori, the fact that (ϵ, π)-symmetry is undecidable does not mean that approximate symmetry for an entire permutation group is undecidable, nor that for fixed ϵ the problem is undecidable. Unfortunately, however, the proof of Theorem 8 uses the permutation group $\mathcal{S}_2$, whose only nontrivial permutation is (1 2). Moreover, the reduction uses the given threshold λ as is, by setting $\lambda = \epsilon$, and the emptiness problem is known to be undecidable even when λ is a fixed number in $(0, 1)$. Thus, we have the following.

▶ **Corollary 9.** *For every $\epsilon \in (0, 1)$, the problem of deciding, given an I/O transducer $\mathcal{T}$ whether $\mathcal{T}$ is (ϵ, π)-symmetric for every $\pi \in \mathcal{S}_k$, is undecidable.*

▶ Remark 10 (Composability). While undecidability of (ϵ, π)-symmetry is unfortunate, the reader may take solace in the fact that (ϵ, π)-symmetry is anyway not preserved under composition. Indeed, if $\mathcal{T}$ is (ϵ, π)-symmetric and (δ, τ)-symmetric, it only guarantees that it is $(\delta + \epsilon, \tau \cdot \pi)$-symmetric. Thus, in order to ensure symmetry over a group, a sound method would have to take into account the *diameter* of the group. This, however, may lose completeness. Thus, (ϵ, π)-symmetry is not a robust notion.

4.2 Parikh Symmetry

The notions of symmetry studied so far have a "letter-by-letter" flavour, where we compare the distribution of specific outputs for a given inputs. We now turn to study a different notion of symmetry, that abstracts away the order of the output symbols, and draws instead on the Parikh image of the computation.

Let $I = \{i_1, \ldots, i_k\}$ and $O = \{o_1, \ldots, o_k\}$. For a word $y = \mathbf{o}_1 \cdots \mathbf{o}_n \in 2^O$, and $1 \leq j \leq k$, define $\#(y, j) = |\{m : o_j \in \mathbf{o}_m\}|$ to be the number of occurrences of o_j in y. Then, we define the *Parikh image*[3] of y to be $\mathfrak{P}(y) = (\#(y, 1), \ldots, \#(y, k)) \in \mathbb{N}^k$.

[3] Observe that this is not the standard Parikh image, in that it is the image with respect to signals in O, rather than to letters in 2^O.

Given a permutation π and a vector $\mathbf{a} = (a_1, \ldots, a_k) \in \mathbb{N}^k$, we define $\pi(\mathbf{a}) = (a_{\pi^{-1}(1)}, \ldots, a_{\pi^{-1}(k)})$. Note that we use π^{-1} so that the following relation holds: if e.g., $\pi(1) = 3$, then index 3 in $\pi(\mathbf{a})$ contains a_1.

Consider an I/O transducer $\mathcal{T}$ and a word $x \in (2^I)^+$. The outputs of $\mathcal{T}$ on x induce a probability measure on (a finite subset of) $\mathbb{N}^k$, where for a vector $\mathbf{a} \in \mathbb{N}^k$ we have $\Pr(\mathfrak{P}(\mathcal{T}(x)) = \mathbf{a}) = \sum_{y:\mathfrak{P}(y)=\mathbf{a}} \Pr(\mathcal{T}(x) = y)$. We can thus also consider the *expected* value of the Parikh image, given by $\mathbb{E}[\mathfrak{P}(\mathcal{T}(x))] = \sum_y \Pr(\mathcal{T}(x) = y)\mathfrak{P}(y)$ (where the product is element-wise, so this is a vector in $\mathbb{N}^k$).

Parikh images give rise to two measures of symmetry: given a permutation π, we say that $\mathcal{T}$ is *π-Parikh distribution symmetric* if for every $x \in (2^I)^+$ and every $\mathbf{a} \in \mathbb{N}^k$ we have $\Pr(\mathfrak{P}(\mathcal{T}(x)) = \mathbf{a}) = \Pr(\mathfrak{P}(\mathcal{T}(\pi(x))) = \pi(a))$. That is, every word x induces the same distribution of Parikh images as $\pi(x)$ does for the permuted images. A weaker notion of symmetry uses expectation: we say that $\mathcal{T}$ is *π-Parikh expected symmetric* if for every $x \in (2^I)^+$ we have $\mathbb{E}[\mathfrak{P}(\mathcal{T}(x))] = \pi(\mathbb{E}[\mathfrak{P}(\mathcal{T}(\pi(x)))])$

Note that Parikh-symmetry assumes the number of occurrences of a certain output signal is meaningful. This is relevant when the output signals measure e.g., number of grants for requests, but makes less sense when the outputs represent e.g., a choice between channels through which a message is routed.

Our algorithmic results about Parikh symmetry use a translation to *probabilistic reward automata* (PRA) [10, Section 5]. A PRA is a PA $\mathcal{A} = \langle Q, \Sigma, \delta, q_0, F\rangle$ equipped with a *reward function* $\mathsf{R} : Q \to \{0,1\}^k$ for some $k \in \mathbb{N}$.[4] The rewards are summed along a run, and the value of a word $w \in \Sigma^*$, denoted $\mathsf{R}(w)$, is the expected reward, that is, the weighted sum of the rewards along all runs, weighted by their respective probabilities. We denote by $\mathcal{A}(w)$ the distribution of reward vectors in $\mathbb{N}^k$, induced by the runs of $\mathcal{A}$ on w.

In order to reason about Parikh images, we propose the following translation.

▶ **Lemma 11.** *Given an I/O trandsucer $\mathcal{T}$, we can construct two PRAs $\mathcal{A}, \mathcal{B}$ over the alphabet 2^I and with reward function of dimension $k = |I|$, such that for every $x \in (2^I)^+$ and for every $\mathbf{a} \in \mathbb{N}^k$, we have that $\Pr(\mathcal{A}(w) = \mathbf{a}) = \Pr(\mathfrak{P}(\mathcal{T}(x)) = \mathbf{a})$, and $\Pr(\mathcal{B}(w) = \mathbf{a}) = \Pr(\mathfrak{P}(\mathcal{T}(\pi(x))) = \pi(\mathbf{a}))$.*

Proof. The translation is similar to the one given in the proof of Theorem 5, where instead of adding 2^O to the alphabet, we collate the Parikh image using the rewards.

Let $\mathcal{T} = \langle I, O, S, s_0, \delta, \boldsymbol{\ell}\rangle$, we construct $\mathcal{A} = \langle S, 2^I, \delta, s_0, S\rangle$ with the following reward function: for every $s \in S$ and $1 \le j \le k$, we have $\mathsf{R}(s)_j = 1$ if $o_j \in \boldsymbol{\ell}(s)$ and $\mathsf{R}(s)_j = 0$ otherwise (that is, $\mathsf{R}(s)$ is the characteristic vector of $\boldsymbol{\ell}(s)$). Thus, $\mathcal{A}$ is identical to $\mathcal{T}$, where we treat all states as accepting, and replace output labels with their characteristic vectors.

The construction of $\mathcal{B}$ is similar, but accounts for the permutation π: we define $\mathcal{B} = \langle S, 2^I, \mu, s_0, S\rangle$ with reward function R', where $\mu(s, \mathbf{i}) = \delta(s, \pi(\mathbf{i}))$ for every state $s \in S$ and $\mathbf{i} \in 2^I$, and $\mathsf{R}'(s) = \pi(\mathsf{R}(s))$ (where R is the reward function of $\mathcal{A}$). It is easy to see that the construction of $\mathcal{A}$ and $\mathcal{B}$ satisfies the conditions of the lemma.

◀

In [10], the problems of distribution-equivalence and expected-equivalence are solved, with complexities NC and RNC, respectively, where NC is the class of problems solvable using circuits of polynomial size and polylogarithmic depth, and RNC is its randomized analogue. It is known that $\mathsf{NC} \subseteq \mathsf{P}$ and $\mathsf{RNC} \subseteq \mathsf{RP}$.

[4] The rewards in [10] also allow -1 rewards, and is set on the transitions of the PRA. Since it is trivial to push rewards from the states to the transitions, our model is simpler.

The distribution-equivalence and expected-equivalence problems, applied to the automata $\mathcal{A}$ and $\mathcal{B}$ obtained as per Lemma 11, exactly correspond to π-distribution symmetry and π-expected symmetry of $\mathcal{T}$, respectively. We thus have the following.

▶ **Theorem 12.** *The problem of deciding, given an I/O transducer $\mathcal{T}$ and a permutation π, whether it is π-Parikh distribution symmetric (resp. π-Parikh expected symmetric), is in* NC *(resp.* RNC*).*

Both notions of Parikh symmetry can be easily shown respect composition, analogously to Lemma 2, in that if $\mathcal{T}$ is both π- and τ- Parikh distribution/expected symmetric, then it is also $\pi \circ \tau$-Parikh distribution/expected symmetric. Thus, we conclude this section with the following.

▶ **Theorem 13.** *The problem of deciding, given an I/O transducer $\mathcal{T}$ and a finite set of generators $X = \{\pi_1, \ldots, \pi_m\}$, whether it is π-Parikh distribution symmetric (resp. π-Parikh expected symmetric) for every $\pi \in \langle X \rangle$, is in* NC *(resp.* RNC*).*

5 Qualitative Symmetry

Section 4.1 rules out a decidable quantitative approximation for symmetry that takes into account the order of the input (at least in the sense of Theorem 8). In lieu of such an approximation, we turn to study a qualitative approximation, whereby we only require that permuting the input does not alter the support of the output distribution.

Let $\mathcal{T}$ be an I/O transducer, and let $\pi \in \mathcal{S}_k$. We say that $\mathcal{T}$ is *π-qualitative-symmetric* if for every $x \in (2^I)^+$ and $y \in (2^O)^+$ we have that $\Pr(\mathcal{T}(x) = y) > 0$ iff $\Pr(\mathcal{T}(\pi(x)) = \pi(y)) > 0$.

Observe that for every x and y as above, $\Pr(\mathcal{T}(x) = y) > 0$ iff there exists a run of $\mathcal{T}$ on x that is labelled y. Thus, in order to study qualitative symmetry, we can ignore the concrete probabilities in $\mathcal{T}$, and only keep information on whether they are positive or not. Therefore, we essentially consider a nondeterministic transducer.

Using a similar translation to that in Theorem 5, but to NFAs instead of PAs, we have the following.

▶ **Lemma 14.** *The problem of deciding, given an I/O transducer $\mathcal{T}$ and a permutation π, whether $\mathcal{T}$ is π-qualitative-symmetric, is in* PSPACE*.*

Proof. Similarly to our approach in Theorem 5, we translate $\mathcal{T}$ to two automata $\mathcal{A}$ and $\mathcal{B}$, where $\mathcal{A}$ mimics the operation of $\mathcal{T}$, and $\mathcal{B}$ works similarly, but under the permutation π. Then, we check the equivalence of $\mathcal{A}$ and $\mathcal{B}$. Instead of using PAs, however, we now use nondeterministic automata (NFAs). An NFA is $\mathcal{N} = \langle Q, \Sigma, \delta, q_0, F \rangle$ where Q is a set of states, Σ is an alphabet, $\delta : Q \times \Sigma \to 2^Q$ is a transition function, q_0 is an initial state, and F are the accepting states. The semantics of NFAs are textbook standard.

Let $\mathcal{T} = \langle I, O, S, s_0, \delta, \boldsymbol{\ell} \rangle$. We define $\mathcal{A} = \langle S, 2^{I \cup O}, \eta, s_0, S \rangle$ and $\mathcal{B} = \langle S, 2^{I \cup O}, \zeta, s_0, S \rangle$, where the transition functions are defined as follows. Let $q \in S$ and $\sigma = \mathbf{i} \cup \mathbf{o}$ with $\mathbf{i} \in 2^I$ and $\mathbf{o} \in 2^O$, then $\eta(q, \sigma) = \{p \in S : \delta(q, \mathbf{i})(p) > 0 \text{ and } \boldsymbol{\ell}(p) = \mathbf{o}\}$ and $\zeta(q, \sigma) = \{p \in S : \delta(q, \pi(\mathbf{i}))(p) > 0 \text{ and } \boldsymbol{\ell}(p) = \pi(\mathbf{o})\}$.

By construction, for every $x \in (2^I)^+$ and $y \in (2^O)^+$ we have that $\Pr(\mathcal{T}(x) = y) > 0$ iff $\mathcal{A}$ accepts $x \otimes y$, and $\Pr(\mathcal{T}(\pi(x)) = \pi(y))$ iff $\mathcal{B}$ accepts $x \otimes y$. Thus, we have that $\mathcal{T}$ is π-qualitative-symmetric iff $L(\mathcal{A}) = L(\mathcal{B})$. Since equivalence of NFAs can be checked in PSPACE, we are done. ◀

We proceed to show a matching lower bound.

▶ **Lemma 15.** *The problem of deciding, given an I/O transducer $\mathcal{T}$ and a permutation π, whether $\mathcal{T}$ is π-qualitative-symmetric, is* PSPACE*-hard.*

Proof. We show the problem is PSPACE-hard via a reduction from the universality problem for NFAs over alphabet $\Sigma = \{0,1\}$ whose states are all accepting. That is, the problem of deciding, given an NFA $\mathcal{A} = \langle Q, \{0,1\}, \delta, q_0, Q\rangle$ (where all states are accepting), whether $L(\mathcal{A}) = \Sigma^*$. This problem was shown to be PSPACE-hard in [9].

The reduction has a similar flavour as that of Theorem 8, in that we use the permutation to switch between components of the transducer. The components themselves, however, are somewhat different.

Let $\mathcal{A} = \langle Q, \{0,1\}, \delta, q_0, Q\rangle$ be an NFA over $\{0,1\}$ with all states accepting. We construct a transducer $\mathcal{T} = \langle I, O, S, s_0, \eta, \boldsymbol{\ell}\rangle$ over $I = \{i_1, i_2\}$ and $O = \{o_1, o_2\}$ as follows. The states are $S = Q \cup \{s_{\text{init}}, s_{\text{mid}}, s_\perp\}$, with the labelling $\boldsymbol{\ell}(q) = \emptyset$ for every $q \in Q$, $\boldsymbol{\ell}(s_{\text{init}}) = \boldsymbol{\ell}(s_{\text{mid}}) = \emptyset$, and $\boldsymbol{\ell}(s_\perp) = \{o_1, o_2\}$. For simplicity, we treat the transition function as nondeterministic $\eta : S \times 2^{I \cup O} \to 2^S$. Technically, this can be thought of as specifying the support of the transition function, with arbitrarily chosen probabilities (e.g., uniform). Note, however, that we do not allow $\emptyset$ in the image of δ, since we must be able to specify probabilities for the transitions. Now, for every $q \in Q$ and $\mathbf{i} \in 2^I$, and we define

$$\eta(q, \mathbf{i}) = \begin{cases} \delta(q,0) \cup \{s_\perp\} & \text{if } \mathbf{i} = \emptyset \\ \delta(q,1) \cup \{s_\perp\} & \text{if } \mathbf{i} = \{i_1, i_2\} \\ \{q_\perp\} & \text{otherwise} \end{cases}$$

That is, within the Q component, we identify $\Sigma = \{0,1\}$ with $\{\emptyset, \{i_1, i_2\}\}$, and whenever there are no corresponding transitions in $\mathcal{A}$, or an "invalid" letter is seen, a transition is taken to $s_\perp$. Note that we add transitions to $s_\perp$ even when there are transition in $\mathcal{A}$, which will play a role later on. The remaining transitions are as follows (see Figure 4).

$\eta(s_{\text{init}}, \{i_1\}) = \{q_0\}$, $\eta(s_{\text{init}}, \{i_2\}) = \{s_{\text{mid}}\}$,
$\eta(s_{\text{init}}, \emptyset) = \eta(s_{\text{init}}, \{i_1, i_2\}) = \{s_\perp\}$, $\eta(s_{\text{mid}}, \emptyset) = \eta(s_{\text{mid}}, \{i_1, i_2\}) = \{s_{\text{mid}}, s_\perp\}$,
$\eta(s_{\text{mid}}, \{i_1\}) = \eta(s_{\text{mid}}, \{i_2\}) = \{s_\perp\}$, and $\eta(s_\perp, \sigma) = \{s_\perp\}$.

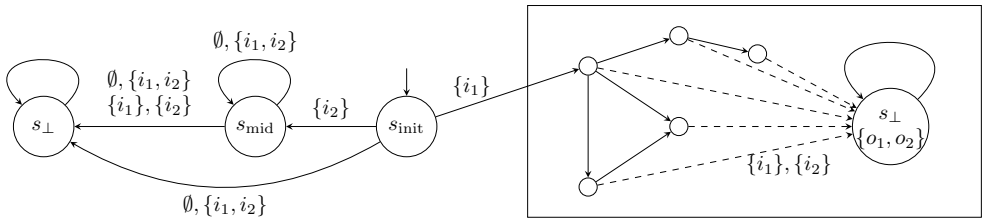


Figure 4 The transducer constructed from an NFA.

Let $\pi = (1\ 2)$. We claim that $L(\mathcal{A}) = \Sigma^*$ iff $\mathcal{T}$ is $(1\ 2)$-qualitative-symmetric.

For the first direction, we prove the contrapositive. Assume $L(\mathcal{A}) \neq \Sigma^*$, and let $w \in \Sigma^* \setminus L(\mathcal{A})$. Keeping our identification of $\Sigma = \{0,1\}$ with $\{\emptyset, \{i_1, i_2\}\}$, consider the word $x = \{i_1\} \cdot w$. Since there are no runs of $\mathcal{A}$ on w, it follows that within the Q component, after reading w, the only reachable state is $s_\perp$. Thus, if $z \in (2^O)^+$ is such that $\Pr(\mathcal{T}(x) = z) > 0$, then z is of the form $\emptyset^+ \cdot \{o_1, o_2\}^+$. In particular, let $y = \emptyset^{|w|+1}$, then $\Pr(\mathcal{T}(x) = y) = 0$. However, a possible run of $\mathcal{T}$ on $\pi(x)$ is $s_{\text{init}}, s_{\text{mid}}^{|w|}$, which induces the labels $y = \pi(y)$. Thus, $\Pr(\mathcal{T}(\pi(x)) = \pi(y)) > 0$, so $\mathcal{T}$ is not π-qualitative-symmetric.

Conversely, assume that $L(\mathcal{A}) = \Sigma^*$, and consider $x \in (2^I)^+$ and $y \in (2^O)^+$. We claim

that $\Pr(\mathcal{T}(x) = y) > 0$ iff $\Pr(\mathcal{T}(\pi(x)) = \pi(y)) > 0$. Observe that similarly to Theorem 8, all the labels on $\mathcal{T}$ are invariant under π, so the above can be stated as

$$\Pr(\mathcal{T}(x) = y) > 0 \text{ iff } \Pr(\mathcal{T}(\pi(x)) = y) > 0. \tag{2}$$

Now, if x starts with either $\emptyset$ or $\{i_1, i_2\}$, then there is a single run on x and on $\pi(x)$, namely $s_{\text{init}}, s_\perp$, so both x and $\pi(x)$ induce the same distribution on output sequences. Thus, Equation (2) holds.

Next, similarly to Theorem 8, we can again assume without loss of generality that x starts with $\{i_1\}$, otherwise we use $\pi(x)$. Thus, x is either of the form $\{i_1\} \cdot w$ or of the form $\{i_1\} \cdot w \cdot \{\{i_1\}, \{i_2\}\} \cdot (2^I)^*$ with $w \in \{\emptyset, \{i_1, i_2\}\}^*$.

In the former case, recall that η follows the transition function of $\mathcal{A}$, as well as allowing at each point to reach $s_\perp$. Thus, $\mathcal{T}(x)$ assigns positive probability to every word of the form $\emptyset^+\{o_1, o_2\}^*$ (of length $|w| + 1$). Observe that $\pi(w) = w$, and hence $\pi(x) = \{i_2\}w$, which induces a distribution with the same support, and again Equation (2) holds.

In the latter case, x is of the form $\{i_1\} \cdot w \cdot \{\{i_1\}, \{i_2\}\} \cdot (2^I)^*$, where upon reading either $\{i_1\}$ or $\{i_2\}$, the runs in the Q component all collapse to $s_\perp$. Thus, the support of $\mathcal{T}(x)$ comprises words of the form $\emptyset^+\{o_1, o_2\}^*$ where the $\emptyset^+$ prefix is at most of length $|w| + 1$. Since $\pi(\{i_1\}) = \{i_2\}$ and $\pi(\{i_2\}) = \{i_1\}$, then by the definition of η, the distribution $\mathcal{T}(\pi(x))$ has the same support (as runs that remain in s_{mid} collapse to $s_\perp$ at the same stage). We thus conclude the claim. Finally, it is easy to see that the reduction is polynomial. ◀

Combining Lemmas 14 and 15, we have the following.

▶ **Theorem 16.** *The problem of deciding, given an I/O transducer $\mathcal{T}$ and a permutation π, whether $\mathcal{T}$ is π-qualitative-symmetric, is* PSPACE*-complete.*

As in Section 4, since we use the permutation group $\mathcal{S}_2$ for our hardness result, we have the following.

▶ **Corollary 17.** *The problem of deciding whether a given I/O transducer $\mathcal{T}$ is π-qualitative-symmetric for every $\pi \in \mathcal{S}_k$ is* PSPACE*-complete.*

6 Extensions and Research Directions

Extensions

The setting considered thus far restricts to corresponding input and output sets of the form $I = \{i_1, \ldots, i_k\}$ and $O = \{o_1, \ldots, o_k\}$. Typically, however, systems also include signals that are not process-specific, such as whether the system is ready, whether there is an error, etc. We can easily incorporate these into the setting. Indeed, adding input signals that are ignored by permutations can be inserted *mutatis-mutandis* to all the automata constructions we use. In addition, the lower bounds trivially carry over.

In addition, some systems have multiple sets of inputs and/or output signals that belong to processes, such as read grants and write grants, both of which are process-specific outputs. Again, our framework can easily be fit with this extension, by permuting each collection of process-specific inputs or outputs separately.

Research Directions

Process symmetry often arises in model checking, and exploiting it correctly can significantly reduce the size of specifications (and hence the time spent in model checking), as well as give insight into the behaviour of the system. In this work, we introduce several variants of process symmetry, and study their algorithmic aspects. Specifically, we show that exact

symmetry can be decided in polynomial time, whereas the approximate version via the L_∞ metric becomes undecidable. A coarser, qualitative approximation, can be decided in PSPACE. In addition, a different type of symmetry, which looks only at the Parikh image of the output, can be decided efficiently.

The notions of symmetry studied in this work restrict to either letter-by-letter symmetry, or Parikh symmetry. However, many other directions can exploit the structure of words as temporal objects to define other symmetry measures. These include *eventual symmetry*, where we require symmetry to take place only after a finite prefix, *sliding-window symmetry*, where we look at Parikh images within a sliding window, while requiring window-by-window symmetry, as well as notions of symmetry that are only relevant for infinite words, such as the limit-average Parikh image.

References

1. Thomas Ball and Orna Kupferman. Vacuity in testing. In *International Conference on Tests and Proofs*, pages 4–17. Springer, 2008.
2. Peter J Cameron et al. *Permutation groups*, volume 45. Cambridge University Press, 1999.
3. Edmund M. Clarke, Reinhard Enders, Thomas Filkorn, and Somesh Jha. Exploiting symmetry in temporal logic model checking. *Formal methods in system design*, 9(1-2):77–104, 1996.
4. Edmund M Clarke Jr, Orna Grumberg, Daniel Kroening, Doron Peled, and Helmut Veith. *Model checking*. MIT press, 2018.
5. A Donaldson and Alice Miller. Symmetry reduction for probabilistic systems. In *Proc. 12th workshop on Automated Reasoning*, pages 17–18, 2005.
6. E Allen Emerson and A Prasad Sistla. Symmetry and model checking. *Formal methods in system design*, 9(1-2):105–131, 1996.
7. Hugo Gimbert and Youssouf Oualhadj. Probabilistic automata on finite words: Decidable and undecidable problems. In *International Colloquium on Automata, Languages, and Programming*, pages 527–538. Springer, 2010.
8. C Norris Ip and David L Dill. Better verification through symmetry. *Formal methods in system design*, 9(1-2):41–75, 1996.
9. Jui-Yi Kao, Narad Rampersad, and Jeffrey Shallit. On nfas where all states are final, initial, or both. *Theoretical Computer Science*, 410(47-49):5010–5021, 2009.
10. Stefan Kiefer and Björn Wachter. Stability and complexity of minimising probabilistic automata. In *International Colloquium on Automata, Languages, and Programming*, pages 268–279. Springer, 2014.
11. Marta Kwiatkowska, Gethin Norman, and David Parker. Symmetry reduction for probabilistic model checking. In *International Conference on Computer Aided Verification*, pages 234–248. Springer, 2006.
12. Anthony W Lin, Truong Khanh Nguyen, Philipp Rümmer, and Jun Sun. Regular symmetry patterns. In *International Conference on Verification, Model Checking, and Abstract Interpretation*, pages 455–475. Springer, 2016.
13. Omid Madani, Steve Hanks, and Anne Condon. On the undecidability of probabilistic planning and related stochastic optimization problems. *Artificial Intelligence*, 147(1-2):5–34, 2003.
14. Azaria Paz. *Introduction to probabilistic automata*. Academic Press, 2014.
15. Marcel Paul Schützenberger. On the definition of a family of automata. *Inf. Control.*, 4(2-3):245–270, 1961.
16. A Prasad Sistla, Viktor Gyuris, and E Allen Emerson. Smc: a symmetry-based model checker for verification of safety and liveness properties. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 9(2):133–166, 2000.
17. Corinna Spermann and Michael Leuschel. Prob gets nauty: Effective symmetry reduction for b and z models. In *2008 2nd IFIP/IEEE International Symposium on Theoretical Aspects of Software Engineering*, pages 15–22. IEEE, 2008.

18 Wen-Guey Tzeng. A polynomial-time algorithm for the equivalence of probabilistic automata. *SIAM Journal on Computing*, 21(2):216–227, 1992.

19 Thomas Wahl and Alastair Donaldson. Replication and abstraction: Symmetry in automated formal verification. *Symmetry*, 2(2):799–847, 2010.