Logical Methods in Computer Science
Volume 19, Issue 4, 2023, pp. 19:1–19:32
https://lmcs.episciences.org/

Submitted Aug. 16, 2022
Published Dec. 05, 2023

SIMULATION BY ROUNDS OF LETTER-TO-LETTER TRANSDUCERS

ANTONIO ABU NASSAR AND SHAULL ALMAGOR

Technion, Israel
e-mail address: antonio@cs.technion.ac.il, shaull@technion.ac.il

Abstract. Letter-to-letter transducers are a standard formalism for modeling reactive systems. Often, two transducers that model similar systems differ locally from one another, by behaving similarly, up to permutations of the input and output letters within "rounds". In this work, we introduce and study notions of simulation by rounds and equivalence by rounds of transducers. In our setting, words are partitioned to consecutive subwords of a fixed length k, called rounds. Then, a transducer $\mathcal{T}_1$ is k-round simulated by transducer $\mathcal{T}_2$ if, intuitively, for every input word x, we can permute the letters within each round in x, such that the output of $\mathcal{T}_2$ on the permuted word is itself a permutation of the output of $\mathcal{T}_1$ on x. Finally, two transducers are k-round equivalent if they simulate each other.

We solve two main decision problems, namely whether $\mathcal{T}_2$ k-round simulates $\mathcal{T}_1$ (1) when k is given as input, and (2) for an existentially quantified k.

We demonstrate the usefulness of the definitions by applying them to process symmetry: a setting in which a permutation in the identifiers of processes in a multi-process system naturally gives rise to two transducers, whose k-round equivalence corresponds to stability against such permutations.

1. Introduction

Reactive systems interact with their environment by receiving inputs, corresponding to the state of the environment, and sending outputs, which describe actions of the system. Finite-state reactive systems are often modeled by *transducers* – finite-state machines over alphabets Σ_I and Σ_O of inputs and outputs, respectively, which read an input letter in Σ_I, and respond with an output in Σ_O. Such transducers are amenable to automatic verification of certain properties (e.g., LTL model-checking), and are therefore useful in practice. Nonetheless, modeling complex systems may result in huge transducers, which makes verification procedures prohibitively expensive, and makes understanding the constructed transducers difficult.

A common approach to gain a better understanding of a transducer (or more generally, any system) is *simulation* [Mil71], whereby a transducer $\mathcal{T}_1$ is simulated by a "simpler" transducer $\mathcal{T}_2$ in such a way that model checking is easier on $\mathcal{T}_2$, and the correctness of the desired property is preserved under the simulation. Usually, "simpler" means smaller,

Key words and phrases: Transducers, Permutations, Parikh, Simulation, Equivalence.
[*] A preliminary version was published in CSL 2022 [AA22].
This research was supported by the ISRAEL SCIENCE FOUNDATION (grant No. 989/22).

as in standard simulation [Mil71] and fair simulation [HKR97], but one can also view e.g., linearization of concurrent programs [HW87] as a form of simulation by a simpler machine.

In this work, we introduce and study new notions of simulation and of equivalence for transducers, based on *rounds*: consider an input word $x \in \Sigma_I^*$ whose length is $k \cdot R$ for some $k, R > 0$. We divide the word into R disjoint infixes of length k, each called a round of w. We then say that two words $x, x' \in \Sigma_I^{kR}$ are k-round equivalent, denoted $x' \asymp_k x$, if x' is obtained from x by permuting the positions of letters within each round of x. For example *abcabc* and *cbaacb* are 3-round equivalent, since *cba* is a permutation of *abc* and so is *acb*. Example 3.1 presents a pair of words that are 3-round equivalent but not 4-round equivalent. We now say that a transducer $\mathcal{T}_1$ is *k-round simulated* by a transducer $\mathcal{T}_2$, denoted $\mathcal{T}_1 \prec_k \mathcal{T}_2$, if for every[1] input $x \in \Sigma_I^{kR}$ we can find $x' \asymp_k x$ such that the outputs of $\mathcal{T}_1$ on x and $\mathcal{T}_2$ on x', denoted y, y' respectively, are also round equivalent: $y' \asymp_k y$. Intuitively, $\mathcal{T}_1 \prec_k \mathcal{T}_2$ means that every behaviour of $\mathcal{T}_1$ is captured by $\mathcal{T}_2$, up to permutations within each round. When we have both $\mathcal{T}_1 \prec_k \mathcal{T}_2$ and $\mathcal{T}_2 \prec_k \mathcal{T}_1$, we say that they are k-round equivalent and denote this by $\mathcal{T}_1 \equiv_k \mathcal{T}_2$.

The benefit of k-round simulation is twofold. First, it may serve as an alternative simulation technique for reducing the state space while maintaining the correctness of certain properties. Second, we argue that k-round simulation is in and of itself a design concern. Indeed, in certain scenarios, as follows, we can naturally design a transducer $\mathcal{T}_2$ that performs a certain task in an ideal, but not realistic, way, and we want to check that an existing design, namely $\mathcal{T}_1$, is simulated by this ideal. In particular, this is useful when dealing with systems that naturally work in rounds, such as schedulers (e.g., Round Robin, cf. Example 3.3), arbiters, and other resource allocation systems.

We now demonstrate both benefits by an example.

Example 1.1. Consider a monitor M for the fairness of a distributed system with 10 processes $\mathcal{P} = \{1, \dots, 10\}$. At each timestep, M receives as input the ID of the process currently working. The monitor then verifies that in each round of 10 steps, every process works exactly once. As long as this holds, the monitor keeps outputting `safe`; otherwise, it outputs `error`.

M can be modeled by a transducer $\mathcal{T}_1$ that keeps track of the set of processes that have worked in the current round. Thus, the transducer has at least 2^{10} states, as it needs to keep track of the subset of processes that have been seen.

It is not hard to see that $\mathcal{T}_1$ is 10-round simulated by an "ideal" transducer $\mathcal{T}_2$ which expects to see the processes in the order $1, \dots, 10$. This transducer needs roughly 10 states, as it only needs to know the index of the next process it expects to see.

Now, suppose we want to verify some correctness property which is invariant to permutations of the processes within each round of length 10, such as "if there is no `error`, then Process 3 works at least once every 20 steps". Then we can verify this against the much smaller $\mathcal{T}_2$.

The notion of k-round simulation arises naturally in the setting of *process symmetry*. There, the input and output alphabets are $\Sigma_I = 2^I$ and $\Sigma_O = 2^O$ respectively, where $I = \{i_1, \dots, i_m\}$ and $O = \{o_1, \dots, o_m\}$ represent signals corresponding to m processes. Process symmetry addresses the scenario where the identifiers of the processes may be scrambled. For example, if the input $\{i_1, i_2\}$ is generated, the system might actually receive an input $\{i_7, i_4\}$. A system exhibits process symmetry if, intuitively, its outputs

[1]Our formal definition allows to also restrict the input to some regular language $\Lambda \subseteq \Sigma_I^*$, see section 3.

are permuted in a similar way to the inputs. Unfortunately, deterministic systems that are process symmetric are extremely naive, as process symmetry is too restrictive for them. While this can be overcome using probabilistic systems, as studied in [Alm20], it is also desirable to find a definition that is suited for deterministic systems. As we show in section 6, k-round simulation provides such a definition.

The main contributions of this work are as follows. We introduce the notion of k-round simulation and k-round equivalence, and define two decision problems pertaining to them: in *fixed round simulation* we need to decide whether $\mathcal{T}_1 \prec_k \mathcal{T}_2$ for a given value of k, and in *existential round simulation* we need to decide whether there exists some value of k for which $\mathcal{T}_1 \prec_k \mathcal{T}_2$ holds. In fact, we consider a somewhat more elaborate setting, by also allowing the inputs to $\mathcal{T}_1$ to be restricted to some regular language Λ. We solve the first problem by reducing it to the containment of two nondeterministic automata. For the second problem, things become considerably more difficult, and the solution requires several constructions, as well as tools such as Presburger arithmetic and Parikh's theorem. In addition, we demonstrate the usefulness of the definitions in relation to process symmetry.

Related Work. Simulation relations between systems are a well studied notion. We refer the reader to [CHVB18, Chapter 13] and references therein for an exposition. The connection of our notion with standard simulation is only up to motivation, as our measure is semantic: it does not directly relate to the state space; instead, it refers to the *behaviour* of the system rather than its structure.

On the technical level, our work is closely related to *commutative automata* [BS73] and *jumping automata* [FPS15, MZ12] — models of automata capable of reading their input in a discontinuous manner, by jumping from one letter to another. Indeed, our notion of round simulation essentially allows the simulating transducer to read the letters within rounds in a discontinuous manner. This similarity is manifested implicitly in section 5.2, where we encounter similar structures as e.g. the commutative closure in [Hof20] (although the analysis here has a different purpose).

Finally, the initial motivation for this work comes from *process symmetry* [Alm20, CEFJ96, ES96, ID96, LNRS16]. We explore the connections in depth in section 6.

Paper Organization. The rest of this work is organized as follows. In section 2 we present some basic definitions used throughout the paper. In section 3 we introduce k-round simulation and equivalence, define the relevant decision problems, and study some fundamental properties of the definitions. In section 4 we solve fixed round simulation, while developing some technical tools and characterizations that are reused later. section 5 is our main technical result, where we develop a solution for existential round simulation. In particular, in section 5.1 we give an overview of the solution, before going through the technical details in section 5.2. In section 5.3 we give lower bounds for the existential setting. In section 6 we use round simulation to obtain a definition of process symmetry for deterministic transducers, along with an algorithm for deciding it. In section 7 we study the mapping between transducers that induces a simulation. In section 8 we study variants of symmetry and simulation, both refining and coarsening the previous notions. Finally, we conclude with some open problems in section 9.

2. Preliminaries

Automata. A *deterministic finite automaton (*DFA*)* is $\mathcal{A} = \langle \Sigma, Q, q_0, \delta, F\rangle$, where Q is a finite set of states, $q_0 \in Q$ is an initial state, $\delta : Q \times \Sigma \to Q$ is a transition function, and $F \subseteq Q$ is the set of accepting states.

The *run* of $\mathcal{A}$ on a word $w = \sigma_0 \cdot \sigma_1 \cdots \sigma_{n-1} \in \Sigma^*$ is a sequence of states $q_0, q_1, \ldots, q_n$ such that $q_{i+1} = \delta(q_i, \sigma_i)$ for all $0 \le i < n$. The run is *accepting* if $q_n \in F$. A word $w \in \Sigma^*$ is *accepted* by $\mathcal{A}$ if the run of $\mathcal{A}$ on w is accepting. The *language* of $\mathcal{A}$, denoted $L(\mathcal{A})$, is the set of words that $\mathcal{A}$ accepts. We also consider *nondeterministic finite automata (*NFA*)*, where $\delta : Q \times \Sigma \to 2^Q$ and there can be multiple initial states. Then, a run of $\mathcal{A}$ on a word $w \in \Sigma^*$ as above is a sequence of states $q_0, q_1, \ldots, q_n$ such that q_0 is an initial state and $q_{i+1} \in \delta(q_i, \sigma_i)$ for all $0 \le i < n$. Analogously to the deterministic setting, the language of $\mathcal{A}$ is the set of words that have an accepting run. We denote by $|\mathcal{A}|$ the number of states of $\mathcal{A}$.

As usual, we denote by δ^* the transition function lifted to words. For states q, q' and $w \in \Sigma^*$, we write $q \xrightarrow{w}_{\mathcal{A}} q'$ if $q' \in \delta^*(q, w)$. That is, if there is a run of $\mathcal{A}$ from q to q' while reading w.

An NFA $\mathcal{A}$ can be viewed as a morphism from Σ^* to the monoid $\mathbb{B}^{Q\times Q}$ of $Q \times Q$ Boolean matrices, where we associate with a letter $\sigma \in \Sigma$ its *type* $\tau_{\mathcal{A}}(\sigma) \in \mathbb{B}^{Q\times Q}$ defined by $(\tau_{\mathcal{A}}(\sigma))_{q,q'} = 1$ if $q \xrightarrow{\sigma}_{\mathcal{A}} q'$, and $(\tau_{\mathcal{A}}(\sigma))_{q,q'} = 0$ otherwise. We lift the definition of types to Σ^* by defining, for a word $w = \sigma_1 \cdots \sigma_n \in \Sigma^*$, its type as $\tau_{\mathcal{A}}(w) = \tau_{\mathcal{A}}(\sigma_1) \cdots \tau_{\mathcal{A}}(\sigma_n)$ where the concatenation denotes Boolean matrix product. It is easy to see that $(\tau_{\mathcal{A}}(w))_{q,q'} = 1$ iff $q \xrightarrow{w}_{\mathcal{A}} q'$. For example, the types of the letters a and b in the automaton in fig. 1 are the 3×3 matrices

$$\tau_{\mathcal{A}}(a) = \begin{array}{c} \\ q_0 \\ q_1 \\ q_2 \end{array} \begin{array}{c} \begin{array}{ccc} q_0 & q_1 & q_2 \end{array} \\ \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \end{array}, \quad \tau_{\mathcal{A}}(b) = \begin{array}{c} \\ q_0 \\ q_1 \\ q_2 \end{array} \begin{array}{c} \begin{array}{ccc} q_0 & q_1 & q_2 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix} \end{array},$$

and the type of the word $w = ab$ in the transducer in fig. 1 is the matrix

$$\tau_{\mathcal{A}}(w) = \begin{array}{c} \\ q_0 \\ q_1 \\ q_2 \end{array} \begin{array}{c} \begin{array}{ccc} q_0 & q_1 & q_2 \end{array} \\ \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \end{array} = \tau_{\mathcal{A}}(a) \cdot \tau_{\mathcal{A}}(b).$$

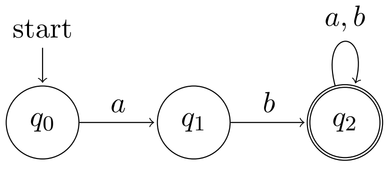


Figure 1: A nondeterministic automaton with one initial state q_0 and one accepting state q_2.

Transducers. Consider two sets Σ_I and Σ_O representing input and output alphabets, respectively. A Σ_I/Σ_O *transducer* is $\mathcal{T} = \langle \Sigma_I, \Sigma_O, Q, q_0, \delta, \boldsymbol{\ell} \rangle$ where Q, $q_0 \in Q$, and $\delta : Q \times \Sigma_I \to Q$ are as in a DFA, and $\boldsymbol{\ell} : Q \to \Sigma_O$ is a labelling function on the states. For a word $w \in \Sigma_I^*$, consider the run $\rho = q_0, \dots, q_n$ of $\mathcal{T}$ on w. We define its output $\boldsymbol{\ell}(\rho) = \boldsymbol{\ell}(q_1) \cdots \boldsymbol{\ell}(q_n) \in \Sigma_O^*$, and we define the output of $\mathcal{T}$ on w to be $\mathcal{T}(w) = \boldsymbol{\ell}(\rho)$. Observe that we ignore the labelling of the initial state in the run, so that the length of the output matches that of the input.

Words and rounds. Consider a word $w = \sigma_0 \cdots \sigma_{n-1} \in \Sigma^*$. We denote its length by $|w|$, and for $0 \le i \le j < |w|$ we define $w[i : j] = \sigma_i \cdots \sigma_j$. For $k > 0$, if $|w| = kR$ for some $R \in \mathbb{N}$, then for every $0 \le r < R$ we refer to $w[rk : r(k+1) - 1]$ as the *r-th round* in w (of length k), and we write $w = \gamma_0 \cdots \gamma_{R-1}$ where γ_r is the r-th round. We emphasize that k indicates the length of each round, not the number of rounds.

In particular, throughout the paper we consider words $(x, y) \in (\Sigma_I^k \times \Sigma_O^k)^*$ and their rounds of length k. In such cases, we sometimes use the natural embedding of $(\Sigma_I^k \times \Sigma_O^k)^*$ in $(\Sigma_I \times \Sigma_O)^*$ and in $\Sigma_I^* \times \Sigma_O^*$, and refer to these sets interchangeably.

Parikh vectors and permutations. Consider an alphabet Σ. For a word $w \in \Sigma^*$ and a letter $\sigma \in \Sigma$, we denote by $\#_\sigma(w)$ the number of occurrences of σ in w. The *Parikh map* $\mathfrak{P} : \Sigma^* \to \mathbb{N}^\Sigma$ maps every word $w \in \Sigma^*$ to a *Parikh vector* $\mathfrak{P}(w) \in \mathbb{N}^\Sigma$, where $\mathfrak{P}(w)(\sigma) = \#_\sigma(w)$. We lift this to languages by defining, for $L \subseteq \Sigma^*$, $\mathfrak{P}(L) = \{\mathfrak{P}(w) : w \in L\}$.

For $\boldsymbol{p} \in \mathbb{N}^\Sigma$ (in the following we consistently denote vectors in $\mathbb{N}^\Sigma$ by bold letters) we write $|\boldsymbol{p}| = \sum_{\sigma \in \Sigma} \boldsymbol{p}(\sigma)$. In particular, for a word $w \in \Sigma^*$ we have $|\mathfrak{P}(w)| = |w|$.

By Parikh's theorem [Par66], for every NFA $\mathcal{A}$ we have that $\mathfrak{P}(L(\mathcal{A}))$ is a *semilinear set* – that is, a finite union of sets of the form $\{\, \boldsymbol{p} + \lambda_1 \boldsymbol{s_1} + \ldots + \lambda_1 \boldsymbol{s_m} \mid \lambda_1, \dots, \lambda_m \in \mathbb{N} \,\}$ where $\boldsymbol{p}, \boldsymbol{s_1}, \dots, \boldsymbol{s_m} \in \mathbb{N}^d$.

Consider words $x, y \in \Sigma^*$. We say that x is a *permutation* of y if $\mathfrak{P}(x) = \mathfrak{P}(y)$ (indeed, in this case y can be obtained from x by permuting its letters). In particular this implies $|x| = |y|$.

3. Round Simulation and Round Equivalence

Consider two k-round words $x, y \in \Sigma^{kR}$ with the same number of rounds R, and denote their rounds by $x = \alpha_0 \cdots \alpha_{R-1}$ and $y = \beta_0 \cdots \beta_{R-1}$. We say that x and y are *k-round equivalent*, denoted $x \asymp_k y$ (or $x \asymp y$, when k is clear from context)[2], if for every $0 \le r < R$ we have that $\mathfrak{P}(\alpha_r) = \mathfrak{P}(\beta_r)$. That is, $x \asymp y$ iff the r-th round of y is a permutation of the r-th round of x, for every r. Indeed, $\asymp$ is an equivalence relation.

Example 3.1 (Round-equivalence for words)**.** Consider the words $x = abaabbabbbaa$ and $y = baabbaabbaba$ over the alphabet $\Sigma = \{a, b\}$. Looking at the words as 3-round words, one can see in table 1 that rounds of length 3 in y are all permutations of those in x, which gives $x \asymp_3 y$. However, looking at the rounds of length 4 of x, y, the number of occurrences of b already in the first round of x and of y is different, so $x \not\asymp_4 y$, as illustrated in table 2.

Let Σ_I and Σ_O be input and output alphabets, let $\Lambda \subseteq \Sigma_I^*$ be a regular language, and let $k > 0$. Consider two Σ_I/Σ_O transducers $\mathcal{T}_1$ and $\mathcal{T}_2$. We say that $\mathcal{T}_2$ *k-round simulates* $\mathcal{T}_1$

[2]Conveniently, our symbol for round equivalence is a rounded equivalence.

Table 1: x and y are 3-round equivalent

x	aba	abb	abb	baa
y	baa	bba	abb	aba

Table 2: x and y are not 4-round equivalent

x	*abaa*	bbab	bbaa
y	*baab*	baab	baba

restricted to Λ, denoted $\mathcal{T}_1 \prec_{k,\Lambda} \mathcal{T}_2$, if for every k-round word $x \in \Lambda$ there exists a k-round word $x' \in \Sigma_I^*$ such that $x \asymp_k x'$ and $\mathcal{T}_1(x) \asymp_k \mathcal{T}_2(x')$.

Intuitively, $\mathcal{T}_1 \prec_{k,\Lambda} \mathcal{T}_2$ if for every input word $x \in \Lambda$, we can permute each round of length k in x to obtain a new word x', such that the outputs of $\mathcal{T}_1$ on x and of $\mathcal{T}_2$ on x' are k-round equivalent. Note that the definition is not symmetric: the input x for $\mathcal{T}_1$ is universally quantified, while x' is chosen according to x. We illustrate this in Example 3.5.

If $\mathcal{T}_1 \prec_{k,\Lambda} \mathcal{T}_2$ and $\mathcal{T}_2 \prec_{k,\Lambda} \mathcal{T}_1$ we say that $\mathcal{T}_1$ and $\mathcal{T}_2$ are *k-round equivalent restricted to* Λ, denoted $\mathcal{T}_1 \equiv_{k,\Lambda} \mathcal{T}_2$. In the special case where $\Lambda = \Sigma_I^*$ (i.e., when we require the simulation to hold for every input), we omit it from the subscript and write $\mathcal{T}_1 \prec_k \mathcal{T}_2$.

Remark 3.2 (On the role of Λ)**.** Transducers have a "universal" flavor, in that every input string is assigned an output. In many settings, however, inputs of interest should comply to some simple form, and are otherwise irrelevant. The restriction language Λ allows the designer to specify that we only care about symmetry when the input is correctly formed.

We note that it is technically easy to add a similar restriction-language for $\mathcal{T}_2$, although we find it less motivated, as $\mathcal{T}_2$ is meant to be an abstraction of $\mathcal{T}_1$ for the purpose of verification, rather than a concrete model to act in an environment.

Example 3.3 (Round Robin)**.** We consider a simple version of the Round Robin scheduler for three processes $\mathcal{P} = \{0, 1, 2\}$. In each time step, the scheduler outputs either a singleton set containing the ID of the process whose request is granted, or an empty set if the process whose turn it is did not make a request. Depending on the ID $i \in \{0, 1, 2\}$ of the first process, we model the scheduler as a $2^{\mathcal{P}}/2^{\mathcal{P}}$ transducer $\mathcal{T}_i = \langle 2^{\mathcal{P}}, 2^{\mathcal{P}}, Q, q_{(i-1)\%3}, \delta, \boldsymbol{\ell}\rangle$ depicted in fig. 2, where % is the mod operator, $Q = \{q_0, q_1, q_2, q'_0, q'_1, q'_2\}$, $\delta(q_i, \sigma) = q_{(i+1)\%3}$ if $i + 1 \in \sigma$ and $\delta(q_i, \sigma) = q'_{(i+1)\%3}$ otherwise, $\boldsymbol{\ell}(q_i) = \{i\}$ and $\boldsymbol{\ell}(q'_i) = \emptyset$.

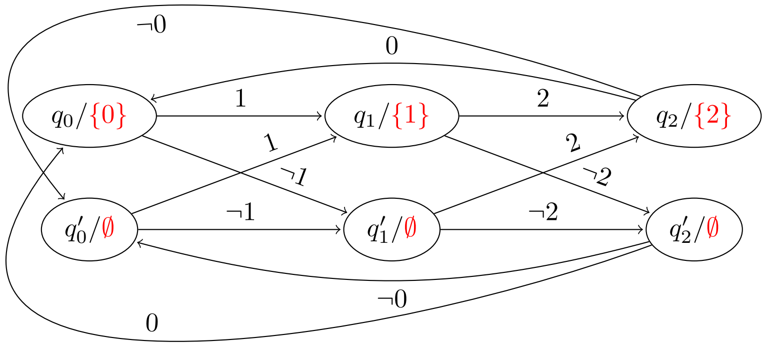


Figure 2: The transducer $\mathcal{T}_i$ for RR, initial state omitted. The input letters σ and $\neg\sigma$ mean all letters from $2^{\mathcal{P}}$ that, respectively, contain or do not contain σ. The labels are written in red.

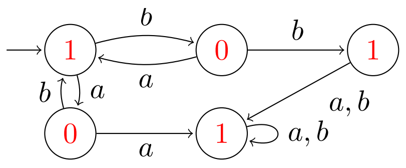
Technically, the initial state changes the behaviour of $\mathcal{T}_i$ significantly (e.g. we have $\mathcal{T}_0(\{0\}\{2\}\{1\}) = \{0\}\emptyset\emptyset$ whereas $\mathcal{T}_1(\{0\}\{2\}\{1\}) = \emptyset\{2\}\emptyset$). Conceptually, however, changing the initial state does not alter the behaviour, as long as the requests are permuted accordingly. This is captured by round equivalence, as follows.

We argue that, if we allow permutation of the input letters, then the set of processes whose requests are granted in each round is independent of the start state. This is equivalent to saying $\mathcal{T}_0 \equiv_3 \mathcal{T}_j$ for $j \in \{1, 2\}$, which indeed holds: if $j = 1$ then we permute all rounds of the form $\sigma_0\sigma_1\sigma_2$ to $\sigma_1\sigma_2\sigma_0$, and similarly if $j = 2$ then we permute all rounds to $\sigma_2\sigma_0\sigma_1$. It is easy to see that the run of $\mathcal{T}_i$ on the permuted input grants outputs that are 3-round equivalent to the output of $\mathcal{T}_0$ on the non-permuted input.

Remark 3.4. In Example 3.3, the constant k of round equivalence is equal to the number of processes $k = 3$. This need not be the case in general. Indeed, one could define Round Robin over 3 processes that follows the request order e.g., 111232332. It is easy to show that in this case, the natural round length is 9, and that permutations of 3-rounds are not enough to reorder inputs starting from different initial states.

In Example 3.3, the transducers satisfied not only round simulation, but also round equivalence. We now show that this is not always the case for simulating transducers.

Example 3.5 (Round simulation is not symmetric)**.** Consider the Σ_I/Σ_O transducers $\mathcal{T}_1$ and $\mathcal{T}_2$ over the alphabet $\Sigma_I = \{a, b\}$ and $\Sigma_O = \{0, 1\}$, depicted in fig. 3. We claim that $\mathcal{T}_1 \prec_2 \mathcal{T}_2$



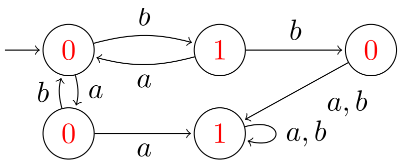


Figure 3: Transducers $\mathcal{T}_1$ (left) and $\mathcal{T}_2$ (right) illustrate the asymmetry in the definition of round equivalence (see Example 3.5).

but $\mathcal{T}_2 \not\prec_2 \mathcal{T}_1$. Starting with the latter, observe that $\mathcal{T}_2(ab) = 00$, but $\mathcal{T}_1(ab) = \mathcal{T}_1(ba) = 01$. Since $00 \not\asymp_2 01$, we have $\mathcal{T}_2 \not\prec_2 \mathcal{T}_1$.

We turn to show that $\mathcal{T}_1 \prec_2 \mathcal{T}_2$. Observe that for every input word of the form $x \in (ab + ba)^m$, we have $\mathcal{T}_1(x) = (01)^m$, and $x \asymp_2 (ba)^m$. So in this case we have that $\mathcal{T}_2((ba)^m) = (10)^m \asymp_2 (01)^m$. Next, for $x \in (ab + ba)^m \cdot bb \cdot w$ for some $w \in \Sigma_I^*$ we have $\mathcal{T}_1(x) = (01)^m 011^{|w|}$ and $x \asymp_2 (ba)^m \cdot bb \cdot w$, for which $\mathcal{T}_2((ba)^m \cdot bb \cdot w) = (01)^m 101^{|w|} \asymp_2 \mathcal{T}_1(x)$. The case where $x \in (ab + ba)^m \cdot aa \cdot w$ is handled similarly. We conclude that $\mathcal{T}_1 \prec_2 \mathcal{T}_2$.

Round simulation and round equivalence give rise to the following decision problems:

- In *fixed round simulation* (resp. *fixed round equivalence*) we are given transducers $\mathcal{T}_1, \mathcal{T}_2$, an NFA for the language Λ, and $k > 0$ in unary, and we need to decide whether $\mathcal{T}_1 \prec_{k,\Lambda} \mathcal{T}_2$ (resp. whether $\mathcal{T}_1 \equiv_{k,\Lambda} \mathcal{T}_2$).
- In *existential round simulation* (resp. *existential round equivalence*) we are given transducers $\mathcal{T}_1, \mathcal{T}_2$ and an NFA for the language Λ, and we need to decide whether there exists $k > 0$ such that $\mathcal{T}_1 \prec_{k,\Lambda} \mathcal{T}_2$ (resp. $\mathcal{T}_1 \equiv_{k,\Lambda} \mathcal{T}_2$).

In the following we identify Λ with an NFA (or DFA) for it, as we do not explicitly rely on its description.

We start by showing that deciding equivalence (both fixed and existential) is reducible, in polynomial time, to the respective simulation problem.

Lemma 3.6. *Fixed (resp. existential) round equivalence is Turing reducible in polynomial time to fixed (resp. existential) round simulation.*

Proof. First, we can clearly reduce fixed round equivalence to fixed round simulation: given an algorithm that decides, given $\mathcal{T}_1, \mathcal{T}_2$, Λ and $k > 0$, whether $\mathcal{T}_1 \prec_{k,\Lambda} \mathcal{T}_2$, we can decide whether $\mathcal{T}_1 \equiv_{k,\Lambda} \mathcal{T}_2$ by using it twice to decide whether both $\mathcal{T}_1 \prec_{k,\Lambda} \mathcal{T}_2$ and $\mathcal{T}_1 \prec_{k,\Lambda} \mathcal{T}_2$ hold.

A slightly more careful examination shows that the same approach can be taken to reduce existential round equivalence to existential round simulation, using the following observation: if $\mathcal{T}_1 \prec_{k,\Lambda} \mathcal{T}_2$, then for every $m \in \mathbb{N}$ it holds that $\mathcal{T}_1 \prec_{mk,\Lambda} \mathcal{T}_2$. Indeed, we can simply group every m rounds of length k and treat them as a single round of length mk.

Now, given an algorithm that decides, given $\mathcal{T}_1, \mathcal{T}_2$ and Λ, whether there exists $k > 0$ such that $\mathcal{T}_1 \prec_{k,\Lambda} \mathcal{T}_2$, we can decide whether $\mathcal{T}_1 \equiv_{k,\Lambda} \mathcal{T}_2$ by using the algorithm twice to decide whether there exists k_1 such that $\mathcal{T}_1 \prec_{k_1,\Lambda} \mathcal{T}_2$ and k_2 such that $\mathcal{T}_2 \prec_{k_2,\Lambda} \mathcal{T}_1$ hold. If there are no such k_1, k_2, then clearly $\mathcal{T}_1 \not\equiv_{k,\Lambda} \mathcal{T}_2$. However, if there are such k_1, k_2, then by the observation above we have $\mathcal{T}_1 \equiv_{k_1 k_2,\Lambda} \mathcal{T}_2$ (we can also take $\mathrm{lcm}(k_1, k_2)$ instead of $k_1 k_2$). $\square$

By Lemma 3.6, for the purpose of upper-bounds, we focus henceforth on round simulation.

4. Deciding Fixed Round Simulation

In this section we show decidability of fixed round simulation (and, by Lemma 3.6, fixed round equivalence). The tools we develop will be used in section 5 to handle the existential variant.

Let Σ_I and Σ_O be input and output alphabets. Consider two Σ_I/Σ_O transducers $\mathcal{T}_1$ and $\mathcal{T}_2$, and let $\Lambda \subseteq \Sigma_I^*$ and $k > 0$. In order to decide whether $\mathcal{T}_1 \prec_{k,\Lambda} \mathcal{T}_2$, we proceed as follows. First, we cast the problem to a problem about deterministic automata. Then, we translate rounds into letters, by working over the alphabets Σ_I^k and Σ_O^k. We construct an NFA, dubbed the *permutation closure*, for each transducer $\mathcal{T}$, that captures the behaviour of $\mathcal{T}$ on words and their permutations. Intuitively, the NFA takes as input a word $(x, y) \in (\Sigma_I^k \times \Sigma_O^k)^*$, guesses a round-equivalent word $x' \asymp x$, and verifies that $\mathcal{T}(x') \asymp \mathcal{T}(x)$. We then show that round simulation amounts to deciding the containment of these NFAs.

We now turn to give the details of the construction of these NFAs.

The trace DFA. Consider a transducer $\mathcal{T} = \langle \Sigma_I, \Sigma_O, Q, q_0, \delta, \boldsymbol{\ell} \rangle$, we define its *trace* DFA $\mathtt{Tr}(\mathcal{T}) = \langle \Sigma_I \times \Sigma_O, Q \cup \{q_\perp\}, q_0, \eta, Q \rangle$ where for $q \in Q$ and $(\sigma, \sigma') \in \Sigma_I \times \Sigma_O$ we define $\eta(q, (\sigma, \sigma')) = \delta(q, \sigma)$ if $\mathcal{T}^q(\sigma) = \sigma'$ and $\eta(q, (\sigma, \sigma')) = q_\perp$ otherwise. $q_\perp$ is a rejecting sink.

$\mathtt{Tr}(\mathcal{T})$ captures the behaviour of $\mathcal{T}$ in that $L(\mathtt{Tr}(\mathcal{T})) = \{ (x, y) \in (\Sigma_I \times \Sigma_O)^* \mid \mathcal{T}(x) = y \}$.

The permutation closure NFA. Consider an NFA $\mathcal{N} = \langle \Sigma_I \times \Sigma_O, S, s_0, \eta, F \rangle$, and let $k > 0$. We obtain from $\mathcal{N}$ an NFA $\mathtt{Perm}_k(\mathcal{N}) = \langle \Sigma_I^k \times \Sigma_O^k, S, s_0, \mu, F \rangle$ where the alphabet is $\Sigma_I^k \times \Sigma_O^k$, and the transition function μ is defined as follows. For a letter $(\alpha, \beta) \in \Sigma_I^k \times \Sigma_O^k$ and a state $s \in S$, we think of (α, β) as a word in $(\Sigma_I \times \Sigma_O)^*$. Then we have

$$\mu(s, (\alpha, \beta)) = \bigcup \left\{ \eta^*(s, (\alpha', \beta')) \mid \mathfrak{P}(\alpha') = \mathfrak{P}(\alpha) \wedge \mathfrak{P}(\beta) = \mathfrak{P}(\beta') \right\}. \tag{4.1}$$

That is, upon reading (α, β), $\mathtt{Perm}_k(\mathcal{N})$ can move to any state s' that is reachable in $\mathcal{N}$ from s by reading a permutation of α, β (denoted α', β'). Recall that for two words

x, x' we have that $x \asymp_k x'$ if for every two corresponding rounds α, α' in x and x' we have $\mathfrak{P}(\alpha) = \mathfrak{P}(\alpha')$. Thus, we have the following.

Observation 4.1. In the notations above, it holds that $L(\mathtt{Perm}_k(\mathcal{N})) = \{(x, y) \in \Sigma_I^* \times \Sigma_O^* \mid \exists x' \asymp_k x, y' \asymp_k y, (x', y') \in L(\mathcal{N}) \wedge \ |x| = |y| = kR$ for some $R \in \mathbb{N}\}$.

Since the transition function of $\mathtt{Perm}_k(\mathcal{N})$ is only defined using permutations of its input letters, we have the following property, which we refer to as *permutation invariance*:

Observation 4.2 (Permutation invariance)**.** For every state $s \in S$ and letters $(\alpha, \beta), (\alpha', \beta') \in \Sigma_I^k \times \Sigma_O^k$, if $\mathfrak{P}(\alpha) = \mathfrak{P}(\alpha')$ and $\mathfrak{P}(\beta) = \mathfrak{P}(\beta')$ then $\mu(s, (\alpha, \beta)) = \mu(s, (\alpha', \beta'))$.

Given a transducer $\mathcal{T}$, we apply the permutation closure to the trace DFA of $\mathcal{T}$. In order to account for the restriction given by $\Lambda \subseteq \Sigma_I^*$, we identify it with $\Lambda \subseteq \Sigma_I^* \times \Sigma_O^*$. Recall that Λ denotes both a language and a corresponding NFA (or DFA), so what this means is that the NFA, reading input from $\Sigma_I^* \times \Sigma_O^*$, simply ignores the second component.

Lemma 4.3. *Consider transducers $\mathcal{T}_1, \mathcal{T}_2$, an* NFA Λ *and* $k > 0$*. Let* $\mathcal{A}_1^k = \mathtt{Perm}_k(\mathtt{Tr}(\mathcal{T}_1) \cap \Lambda)$ *(where the intersection implies the product* NFA *construction) and* $\mathcal{A}_2^k = \mathtt{Perm}_k(\mathtt{Tr}(\mathcal{T}_2))$*, then*

$$L(\mathcal{A}_1^k) = \{\, (x, y) \in \Sigma_I^* \times \Sigma_O^* \mid \exists x' \asymp_k x,\ \mathcal{T}_1(x') \asymp_k y \ \wedge\ |x| = |y| = kR \text{ where } R \in \mathbb{N} \wedge x' \in \Lambda \,\},$$

$$L(\mathcal{A}_2^k) = \{\, (x, y) \in \Sigma_I^* \times \Sigma_O^* \mid \exists x' \asymp_k x,\ \mathcal{T}_2(x') \asymp_k y \ \wedge\ |x| = |y| = kR \text{ where } R \in \mathbb{N} \,\}.$$

Proof. Recall that $\mathtt{Tr}(\mathcal{T})$ accepts a word (x', y') iff $\mathcal{T}(x') = y'$. The claim then follows from Observation 4.1, by replacing the expression $y \asymp y' \wedge (x', y') \in L(\mathtt{Tr}(\mathcal{T}))$ with the equivalent expression $\mathcal{T}(x') \asymp_k y$. □

We now reduce round simulation to the containment of permutation closure NFAs.

Lemma 4.4. *Consider transducers $\mathcal{T}_1, \mathcal{T}_2$, an* NFA Λ *and* $k > 0$*. Let* $\mathcal{A}_1^k = \mathtt{Perm}_k(\mathtt{Tr}(\mathcal{T}_1) \cap \Lambda)$ *and* $\mathcal{A}_2^k = \mathtt{Perm}_k(\mathtt{Tr}(\mathcal{T}_2))$*, then $\mathcal{T}_1 \prec_{k,\Lambda} \mathcal{T}_2$ iff $L(\mathcal{A}_1^k) \subseteq L(\mathcal{A}_2^k)$.*

Proof. For the first direction, assume $\mathcal{T}_1 \prec_{k,\Lambda} \mathcal{T}_2$, and let $(x, y) \in L(\mathcal{A}_1^k)$. By Lemma 4.3, x and y are k-round words, and there exists a word $x' \in \Lambda$ such that $x \asymp x'$ and $\mathcal{T}_1(x') \asymp y$. Since $\mathcal{T}_1 \prec_{k,\Lambda} \mathcal{T}_2$, then applying the definition on x' yields that there exists a k-round word x'' such that $x' \asymp x''$ and such that $\mathcal{T}_1(x') \asymp \mathcal{T}_2(x'')$. Since $\asymp$ is an equivalence relation, it follows that $x \asymp x''$ and $\mathcal{T}_2(x'') \asymp y$, so again by Lemma 4.3 we have $(x, y) \in L(\mathcal{A}_2^k)$.

Conversely, assume $L(\mathcal{A}_1^k) \subseteq L(\mathcal{A}_2^k)$, we wish to prove that for every k-round word $x \in \Lambda$ there exists a word x' such that $x \asymp x'$ and $\mathcal{T}_1(x) \asymp \mathcal{T}_2(x')$. Let $x \in \Lambda$ be a k-round word, and let $y = \mathcal{T}_1(x)$, then clearly $(x, y) \in L(\mathcal{A}_1^k) \subseteq L(\mathcal{A}_2^k)$ (since $x \asymp x$, $\mathcal{T}_1(x) = y \asymp y$ and $x \in \Lambda$). By Lemma 4.3, there exists x' such that $x \asymp x'$ and $\mathcal{T}_2(x') \asymp y = \mathcal{T}_1(x)$, so $\mathcal{T}_2(x') \asymp \mathcal{T}_1(x)$, thus concluding the proof. □

Remark 4.5. The proof of Lemma 4.4 does not require taking the permutation closure of $\mathtt{Tr}(\mathcal{T}_1) \cap \Lambda$, and it could be simplified by using instead of $\mathcal{A}_1^k$, the augmentation of $\mathtt{Tr}(\mathcal{T}_1) \cap \Lambda$ to k-round words. However, such an NFA is not permutation invariant, which is key to our solution for existential round simulation. Since this simplification does not reduce the overall complexity, we use a uniform setting for both solutions.

Lemma 4.4 shows that deciding fixed round equivalence amounts to deciding containment of NFAs. By analyzing the size of the NFAs, we obtain the following.

Theorem 4.6. *Given transducers $\mathcal{T}_1, \mathcal{T}_2$, an* NFA Λ*, and* $k > 0$ *in unary, the problem of deciding whether $\mathcal{T}_1 \prec_{k,\Lambda} \mathcal{T}_2$ is in PSPACE.*

Proof. Let $\mathcal{A}_1^k = \texttt{Perm}_k(\texttt{Tr}(\mathcal{T}_1) \cap \Lambda)$ and $\mathcal{A}_2^k = \texttt{Perm}_k(\texttt{Tr}(\mathcal{T}_2))$. By Lemma 4.4, deciding whether $\mathcal{T}_1 \prec_{k,\Lambda} \mathcal{T}_2$ amounts to deciding whether $L(\mathcal{A}_1^k) \subseteq L(\mathcal{A}_2^k)$. Looking at the dual problem, recall that for two NFAs $\mathcal{N}_1, \mathcal{N}_2$ we have that $L(\mathcal{N}_1) \not\subseteq L(\mathcal{N}_2)$ iff there exists $w \in L(\mathcal{N}_2) \setminus L(\mathcal{N}_1)$ with $|w| \leq |\mathcal{N}_1| \cdot 2^{|\mathcal{N}_2|}$ (this follows immediately by bounding the size of an NFA for $L(\mathcal{N}_1) \cap \overline{L(\mathcal{N}_2)}$). Thus, we can decide whether $L(\mathcal{A}_1^k) \subseteq L(\mathcal{A}_2^k)$ by guessing a word w over $\Sigma_I^k \times \Sigma_O^k$ of single-exponential length (in the size of $\mathcal{A}_1^k$ and $\mathcal{A}_2^k$), and verifying that it is accepted by $\mathcal{A}_1^k$ and not by $\mathcal{A}_2^k$.

Observe that to this end, we do not explicitly construct $\mathcal{A}_1^k$ nor $\mathcal{A}_2^k$, as their alphabet size is exponential. Rather, we evaluate them on each letter of w based on their construction from $\mathcal{T}$. At each step we keep track of a counter for the length of w, a state of $\mathcal{A}_1^k$, and a set of states of $\mathcal{A}_2^k$. Since the number of states in $\mathcal{A}_1^k$ and $\mathcal{A}_2^k$ is the same as that of $\mathcal{T}_1$ and $\mathcal{T}_2$, this requires polynomial space.

By Savitch's theorem we have that coNPSPACE = PSPACE, and the proof is concluded. □

We now establish a PSPACE-hardness lower bound, thus concluding that the problem is PSPACE-complete. In fact, we show a lower bound for round equivalence. Note that a priori, this does not entail a lower bound for round simulation by Lemma 3.6, since the reduction there is a Turing reduction. However, our PSPACE-hardness proof actually explicitly shows the hardness of both simulation and equivalence.

Theorem 4.7. *The problem of deciding, given transducers $\mathcal{T}_1, \mathcal{T}_2$, whether $\mathcal{T}_1 \equiv_{k,\Lambda} \mathcal{T}_2$, is PSPACE-hard, even for $k = 2$ and Λ of constant size (given as a 4-state* DFA*).*

Proof sketch. We show a reduction from the universality problem for NFAs over alphabet $\{0, 1\}$ where all states are accepting and the degree of nondeterminism is at most 2. See appendix A for a proof of PSPACE-hardness of this problem and for the full reduction.

Consider an NFA $\mathcal{N} = \langle Q, \{0, 1\}, \delta, q_0, Q\rangle$ where $|\delta(q, \sigma)| \leq 2$ for every $q \in Q$ and $\sigma \in \{0, 1\}$. Set $\Lambda = (ab + cd)^*$. We construct two transducers $\mathcal{T}_1$ and $\mathcal{T}_2$ over input and output alphabets $\Sigma_I = \{a, b, c, d\}$ and $\Sigma_O = \{\top, \bot\}$ such that $L(\mathcal{N}) = \{0, 1\}^*$ iff $\mathcal{T}_1 \equiv_{2,\Lambda} \mathcal{T}_2$.

Intuitively, our reduction encodes $\{0, 1\}$ over $\{a, b, c, d\}$ by identifying 0 with ab and with ba, and 1 with cd and with dc. Then, $\mathcal{T}_1$ keeps outputting $\top$ for all inputs in Λ, thus mimicking a universal language in $\{0, 1\}^*$ (see fig. 14), whereas $\mathcal{T}_2$ is obtained by replacing every nondeterministic transition of $\mathcal{N}$ on e.g. 0 by two deterministic branches, on e.g. ab and ba (see fig. 15). Hence, when we are allowed to permute ab and ba by round equivalence, we capture the nondeterminism of $\mathcal{N}$.

We show that $L(\mathcal{N}) = \{0, 1\}^*$ iff $\mathcal{T}_1 \equiv_{2,\Lambda} \mathcal{T}_2$ by showing that permuting a word $w \in \Lambda$ essentially amounts to choosing an accepting run of $\mathcal{N}$ on the corresponding word in $\{0, 1\}^*$. □

Corollary 4.8. *Given transducers $\mathcal{T}_1, \mathcal{T}_2$, an* NFA *$\Lambda$, and $k > 0$ in unary, the problem of deciding whether $\mathcal{T}_1 \prec_{k,\Lambda} \mathcal{T}_2$ is PSPACE-complete.*

5. Deciding Existential Round Simulation

In section 4, we established a method for deciding k-round simulation for a given k. This case is for when the systems in question exhibit an apparent symmetry with a round length that a developer can guess; such as Round Robin where the round length is the number of processes involved. However, k is not necessarily given in the general sense.

We turn to solve existential round simulation. That is, given $\mathcal{T}_1, \mathcal{T}_2$ and Λ, we wish to decide whether there exists $k > 0$ such that $\mathcal{T}_1 \prec_{k,\Lambda} \mathcal{T}_2$. By Lemma 4.4, this is equivalent to deciding whether there exists $k > 0$ such that $L(\mathcal{A}_1^k) \subseteq L(\mathcal{A}_2^k)$, as defined therein.

Recall that solving the decision problems of round simulation will aid us in solving the initial problem of round symmetry, which gave the motivation for this work. The transition between the problems is explained in section 6.

5.1. **Intuitive Overview.** We start with an intuitive explanation of the solution and its challenges. For simplicity, assume for now $\Lambda = \Sigma_I^*$, so it can be ignored. The overall approach is to present a practical method for hunting k: in Theorem 5.1, the main result of this section, we give an upper bound on the minimal $k > 0$ for which $\mathcal{T}_1 \prec_k \mathcal{T}_2$, rendering the search space finite. In order to obtain this bound, we proceed as follows. Observe that for a transducer $\mathcal{T}$ and for $0 < k \neq k'$ the corresponding permutation closure NFAs $\texttt{Perm}_k(\texttt{Tr}(\mathcal{T}))$ and $\texttt{Perm}_{k'}(\texttt{Tr}(\mathcal{T}))$ are defined on the same state space, but differ by their alphabet ($\Sigma_I^k \times \Sigma_O^k$ vs $\Sigma_I^{k'} \times \Sigma_O^{k'}$). Thus, by definition, these NFAs obtained from an increasing round length form infinitely many distinct automata. Nonetheless, there are only finitely many possible types of letters (indeed, at most $|\mathbb{B}^{Q\times Q}| = 2^{|Q|^2}$). Therefore, there are only finitely many *type profiles* for NFAs– that is, the set of letter types occurring in the NFA– up to multiplicities of the letter types.

Recall that by Lemma 4.4, we have $\mathcal{T}_1 \prec_k \mathcal{T}_2$ iff $L(\texttt{Perm}_k(\texttt{Tr}(\mathcal{T}_1))) \subseteq L(\texttt{Perm}_k(\texttt{Tr}(\mathcal{T}_2)))$. Intuitively, one could hope that if $\texttt{Perm}_k(\texttt{Tr}(\mathcal{T}_i))$ and $\texttt{Perm}_{k'}(\texttt{Tr}(\mathcal{T}_i))$ have the same type profile, for each $i \in \{1,2\}$, then it holds that $L(\texttt{Perm}_k(\texttt{Tr}(\mathcal{T}_1))) \subseteq L(\texttt{Perm}_k(\texttt{Tr}(\mathcal{T}_2)))$ iff $L(\texttt{Perm}_{k'}(\texttt{Tr}(\mathcal{T}_1))) \subseteq L(\texttt{Perm}_{k'}(\texttt{Tr}(\mathcal{T}_2)))$. Then, if one can bound the index k after which no further type profiles are encountered, then the problem reduces to checking a finite number of containments.

Unfortunately, this is not the case, the reason being that the mapping of letters induced by the equal type profiles $\texttt{Perm}_k(\texttt{Tr}(\mathcal{T}_1))$ and $\texttt{Perm}_{k'}(\texttt{Tr}(\mathcal{T}_1))$ may differ from the mapping induced by $\texttt{Perm}_k(\texttt{Tr}(\mathcal{T}_2))$ and $\texttt{Perm}_{k'}(\texttt{Tr}(\mathcal{T}_2))$, and thus one cannot translate language containment between the two pairs. We overcome this difficulty, however, by working from the start with product automata that capture the structure of both $\mathcal{T}_1$ and $\mathcal{T}_2$ simultaneously, and thus unify the letter mapping. We dub them *redundant product automata* for their apparent redundancy.

We are now left with the problem of bounding the minimal k after which no new type profiles appear. In order to provide this bound, we show that for every type profile, the set of indices in which it occurs is semilinear. Then, by finding a bound for each type profile, we obtain the overall bound. The main result of this section is the following.

Theorem 5.1. *Given transducers $\mathcal{T}_1, \mathcal{T}_2$ and Λ, we can effectively compute $K_0 > 0$ such that if $\mathcal{T}_1 \prec_{k,\Lambda} \mathcal{T}_2$ for some $k \in \mathbb{N}$, then $\mathcal{T}_1 \prec_{k',\Lambda} \mathcal{T}_2$ for some $k' \leq K_0$.*

Which by Lemma 4.4 immediately entails the following.

Corollary 5.2. *Existential round simulation is decidable.*

We prove Theorem 5.1 in section 5.2, organized as follows. We start by lifting the definition of types in an NFA to Parikh vectors, and show how these relate to the NFA (in Lemma 5.3). We then introduce Presburger arithmetic and its relation to Parikh's theorem. In Lemma 5.4 we show that the set of Parikh vectors that share a type τ is definable in Presburger arithmetic, which provides the first main step towards our bound.

We then proceed to define the redundant product automata mentioned above, which serve to unify the types between $\mathcal{T}_1$ and $\mathcal{T}_2$. In Observations 5.5 and 5.6 we formalize the connection of these products to the transducers $\mathcal{T}_1$ and $\mathcal{T}_2$. Then, we formally define the type profiles and prove in Lemma 5.7 that they exhibit a semilinear behaviour. Finally, in Lemma 5.8 we prove that when two redundant product automata have the same type profile, then the containment mentioned above can be shown. Combining these results, we obtain Theorem 5.1. A flow diagram for the proof is illustrated in fig. 4.

$$\begin{tikzcd}
\mathcal{D}_1 = \mathtt{Tr}(\mathcal{T}_1) \cap \Lambda \arrow[dr] & & \mathcal{D}_2 = \mathtt{Tr}(\mathcal{T}_2) \arrow[dl] & \text{trace DFA} \\
& \mathcal{B}_1, \mathcal{B}_2 = \mathcal{D}_1 \times \mathcal{D}_2 \arrow[d] & & \text{Redundant products} \\
& \tau_{\mathcal{B}_1}(\mathbf{p}, \mathbf{o}) = \tau_{\mathcal{B}_2}(\mathbf{p}, \mathbf{o}) \arrow[d] & & \text{Types of Parikh vectors} \\
& \Upsilon(\mathcal{B}_1, k) = \Upsilon(\mathcal{B}_2, k) \arrow[d] & & \text{Equivalence of Type Profiles} \\
& \Theta_T(k) \arrow[d] & & \text{PA Formula for Types} \\
& \text{If } \Theta_T(k) \wedge \Theta_T(k') \text{ then } \mathcal{T}_1 \prec_k \mathcal{T}_2 \text{ iff } \mathcal{T}_1 \prec_{k'} \mathcal{T}_2 & &
\end{tikzcd}$$

Figure 4: A flow diagram for the proof steps in section 5.2.

5.2. **Proof of Theorem 5.1.**

Type matrices of Parikh vectors. Consider the alphabet $\Sigma_I^k \times \Sigma_O^k$ for some $k > 0$. Recall that by Observation 4.2, permutation closure NFAs are permutation invariant, and from section 2, the type of a word in an NFA is the transition matrix it induces. In particular, for permutation invariant NFAs, two letters $(\alpha, \beta), (\alpha', \beta') \in \Sigma_I^k \times \Sigma_O^k$ with $\mathfrak{P}(\alpha) = \mathfrak{P}(\alpha')$ and $\mathfrak{P}(\beta) = \mathfrak{P}(\beta')$ have the same type.

Following this, we now lift the definition of types to Parikh vectors. Consider an NFA $\mathcal{N} = \langle \Sigma_I \times \Sigma_O, S, s_0, \eta, F\rangle$, and let $\boldsymbol{p} \in \mathbb{N}^{\Sigma_I}, \boldsymbol{o} \in \mathbb{N}^{\Sigma_O}$ be Parikh vectors with $|\boldsymbol{p}| = |\boldsymbol{o}| = k$. We define the type $\tau_{\mathcal{N}}(\boldsymbol{p}, \boldsymbol{o}) \in \mathbb{B}^{S \times S}$ to be $\tau_{\mathtt{Perm}_k(\mathcal{N})}(\alpha, \beta)$ where $(\alpha, \beta) \in \Sigma_I^k \times \Sigma_O^k$ are such that $\mathfrak{P}(\alpha) = \boldsymbol{p}$ and $\mathfrak{P}(\beta) = \boldsymbol{o}$. By permutation invariance, this is well-defined, i.e. is independent of the choice of α and β.

Note that we use different automata to extract the type of words of different lengths. We obtain a more uniform description as follows.

Lemma 5.3. *In the notations above, for every $s_1, s_2 \in S$, we have $(\tau_{\mathcal{N}}(\boldsymbol{p}, \boldsymbol{o}))_{s_1, s_2} = 1$ iff there exists $(\alpha, \beta) \in \Sigma_I^k \times \Sigma_O^k$ with $\mathfrak{P}(\alpha) = \boldsymbol{p}$ and $\mathfrak{P}(\beta) = \boldsymbol{o}$ such that $s_1 \xrightarrow{(\alpha,\beta)}_{\mathtt{Perm}_k(\mathcal{N})} s_2$.*

Proof. By the definitions preceding the lemma, we have that $\tau_{\mathcal{N}}(\boldsymbol{p}, \boldsymbol{o}) = \tau_{\mathtt{Perm}_k(\mathcal{N})}(\alpha', \beta')$ for some $(\alpha', \beta') \in \Sigma_I^k \times \Sigma_O^k$ are such that $\mathfrak{P}(\alpha') = \boldsymbol{p}$ and $\mathfrak{P}(\beta') = \boldsymbol{o}$. According to the transition function of $\mathtt{Perm}_k(\mathcal{N})$ (as defined in section 4), for every $s_1, s_2 \in S$ we have that $s_1 \xrightarrow{(\alpha',\beta')}_{\mathtt{Perm}_k(\mathcal{N})} s_2$ iff there exist $(\alpha, \beta) \in \Sigma_I^k \times \Sigma_O^k$ with $\mathfrak{P}(\alpha) = \mathfrak{P}(\alpha') = \boldsymbol{p}$ and

$\mathfrak{P}(\beta) = \mathfrak{P}(\beta') = \boldsymbol{o}$ such that $s_1 \xrightarrow{(\alpha,\beta)}_{\mathcal{N}} s_2$. Since the type encodes the reachable pairs of states, this concludes the proof. □

Presburger arithmetic. The first ingredient in the proof of Theorem 5.1 is to characterize the set of Parikh vectors whose type is some fixed matrix $\tau \in \mathbb{B}^{Q\times Q}$. For this characterization, we employ the first-order theory of the naturals with addition and order $\mathrm{Th}(\mathbb{N}, 0, 1, +, <, =)$, commonly known as *Presburger arithmetic (PA)*. We do not give a full exposition of PA but refer the reader to [Haa18] (and references therein) for a survey. In the following we briefly cite the results we need.

For our purposes, a PA formula $\varphi(x_1, \dots, x_d)$, where $x_1, \dots, x_d$ are free variables, is evaluated over $\mathbb{N}^d$, and *defines* the set $\{ (a_1, \dots, a_d) \in \mathbb{N}^d \mid (a_1, \dots, a_d) \models \varphi(x_1, \dots, x_d) \}$. For example, the formula $\varphi(x_1, x_2) := x_1 < x_2 \wedge \exists y.\ x_1 = 2y$ defines the set $\{ (a, b) \in \mathbb{N}^2 \mid a < b \wedge a \text{ is even} \}$.

A fundamental result about PA is that the definable sets in PA are exactly the semilinear sets. In particular, Parikh's theorem states that for every NFA $\mathcal{A}$, $\mathfrak{P}(L(\mathcal{A}))$ is PA definable. In fact, by [VSS05], one can efficiently construct a linear-sized existential PA formula for $\mathfrak{P}(L(\mathcal{A}))$. We can now show that the set of Parikh vectors whose type is τ is PA definable.

Lemma 5.4. *Consider an* NFA $\mathcal{N} = \langle \Sigma_I \times \Sigma_O, S, s_0, \eta, F\rangle$, *and a type* $\tau \in \mathbb{B}^{S\times S}$, *then the set* $\{ (\boldsymbol{p}, \boldsymbol{o}) \in \mathbb{N}^{\Sigma_I} \times \mathbb{N}^{\Sigma_O} \mid \tau_{\mathcal{N}}(\boldsymbol{p}, \boldsymbol{o}) = \tau \}$ *is* PA *definable.*

Proof. Let $\tau \in \mathbb{B}^{S\times S}$, and consider a Parikh vector $(\boldsymbol{p}, \boldsymbol{o}) \in \mathbb{N}^{\Sigma_I} \times \mathbb{N}^{\Sigma_O}$ with $k = |\boldsymbol{p}| = |\boldsymbol{o}|$. By Lemma 5.3, we have that $\tau_{\mathcal{N}}(\boldsymbol{p}, \boldsymbol{o}) = \tau$ iff the following holds for every $s_1, s_2 \in S$: we have $\tau_{s_1,s_2} = 1$ iff there exists a letter $(\alpha, \beta) \in \Sigma_I^k \times \Sigma_O^k$ such that $\mathfrak{P}(\alpha) = \boldsymbol{p}, \mathfrak{P}(\beta) = \boldsymbol{o}$, and $s_1 \xrightarrow{(\alpha,\beta)}_{\mathcal{N}} s_2$.

Consider $s_1, s_2 \in S$ and define $\mathcal{N}^{s_1}_{s_2}$ to be the NFA obtained from $\mathcal{N}$ by setting the initial state to be s_1 and a single accepting state s_2. Then, we have $s_1 \xrightarrow{(\alpha,\beta)}_{\mathcal{N}} s_2$ iff $(\alpha, \beta) \in L(\mathcal{N}^{s_1}_{s_2})$.

Thus, $\tau_{\mathcal{N}}(\boldsymbol{p}, \boldsymbol{o}) = \tau$ iff for every $s_1, s_2 \in S$ we have that $\tau_{s_1,s_2} = 1$ iff there exists a word (α, β) with $\mathfrak{P}(\alpha') = \boldsymbol{p}$ and $\mathfrak{P}(\beta') = \boldsymbol{o}$ such that $(\alpha, \beta) \in L(\mathcal{N}^{s_1}_{s_2})$. Equivalently, we have $\tau_{\mathcal{N}}(\boldsymbol{p}, \boldsymbol{o}) = \tau$ iff for every $s_1, s_2 \in S$ it holds that $\tau_{s_1,s_2} = 1$ iff $(\boldsymbol{p}, \boldsymbol{o}) \in \mathfrak{P}(L(\mathcal{N}^{s_1}_{s_2}))$.

By Parikh's theorem, for every $s_1, s_2 \in S$ we can compute a PA formula ψ_{s_1,s_2} such that $(\boldsymbol{p}, \boldsymbol{o}) \models \psi_{s_1,s_2}$ iff $(\boldsymbol{p}, \boldsymbol{o}) \in \mathfrak{P}(L(\mathcal{N}^{s_1}_{s_2}))$. Now we can construct a PA formula Ψ_τ such that $\tau_{\mathcal{N}}(\boldsymbol{p}, \boldsymbol{o}) = \tau$ iff $(\boldsymbol{p}, \boldsymbol{o}) \models \Psi_\tau$, as follows:

$$\Psi_\tau := \bigwedge_{s_1,s_2\,:\,\tau_{s_1,s_2}=1} \psi_{s_1,s_2} \wedge \bigwedge_{s_1,s_2\,:\,\tau_{s_1,s_2}=0} \neg\psi_{s_1,s_2}.$$

Finally, observe that Ψ_τ defines the set in the premise of the lemma, so we are done. □

The redundant product construction. As mentioned in section 5.1, for the remainder of the proof we want to reason about the types of $\mathtt{Perm}_k(\mathtt{Tr}(\mathcal{T}_1) \cap \Lambda)$ and $\mathtt{Perm}_k(\mathtt{Tr}(\mathcal{T}_2))$ simultaneously. In order to do so, we present an auxiliary product construction.

Let $\mathcal{T}_1, \mathcal{T}_2$ be transducers, $\Lambda \subseteq \Sigma_I^*$ be given by an NFA, and let $\mathcal{D}_1 = \mathtt{Tr}(\mathcal{T}_1) \cap \Lambda$ and $\mathcal{D}_2 = \mathtt{Tr}(\mathcal{T}_2)$. We now consider the product automaton of $\mathcal{D}_1$ and $\mathcal{D}_2$, and endow it with two different acceptance conditions, capturing that of $\mathcal{D}_1$ and $\mathcal{D}_2$, respectively. Formally, for $i \in \{1, 2\}$, denote $\mathcal{D}_i = \langle \Sigma_I \times \Sigma_O, S_i, s^i_0, \eta_i, F_i\rangle$, then the product automaton is defined as $\mathcal{B}_i = \langle \Sigma_I \times \Sigma_O, S_1 \times S_2, (s^1_0, s^2_0), \eta_1 \times \eta_2, G_i\rangle$, where $G_1 = F_1 \times Q_2$ and $G_2 = Q_1 \times F_2$, and $\eta_1 \times \eta_2$ denotes the standard product transition function, namely $\eta_1 \times \eta_2((s_1, s_2), (\sigma, \sigma')) =$

$(\eta_1(s_1,(\sigma,\sigma')),\eta_2(s_2,(\sigma,\sigma')))$. Thus, $\mathcal{B}_i$ tracks both $\mathcal{D}_1$ and $\mathcal{D}_2$, but has the same acceptance condition as $\mathcal{D}_i$. This seemingly "redundant" product construction has the following important properties, which are crucial for our proof:

Observation 5.5. In the notations above, we have the following:

(1) $L(\mathcal{B}_1) = L(\mathcal{D}_1)$ and $L(\mathcal{B}_2) = L(\mathcal{D}_2)$.
(2) For every letter $(\sigma,\sigma') \in \Sigma_I \times \Sigma_O$, we have $\tau_{\mathcal{B}_1}(\sigma,\sigma') = \tau_{\mathcal{B}_2}(\sigma,\sigma')$.

Indeed, Item 1 follows directly from the acceptance condition, and Item 2 is due to the identical transition function of $\mathcal{B}_1$ and $\mathcal{B}_2$.

By Observation 4.1, $L(\texttt{Perm}_k(\mathcal{D}_i))$ depends only on $L(\mathcal{D}_i)$. We thus have the following.

Observation 5.6. The following holds for every $k > 0$:

(1) $L(\texttt{Perm}_k(\mathcal{B}_1)) = L(\texttt{Perm}_k(\texttt{Tr}(\mathcal{T}_1) \cap \Lambda))$.
(2) $L(\texttt{Perm}_k(\mathcal{B}_2)) = L(\texttt{Perm}_k(\texttt{Tr}(\mathcal{T}_2)))$.

Type profiles. We now consider the set of types induced by the redundant product automata $\mathcal{B}_1$ and $\mathcal{B}_2$ on Parikh vectors of words of length k. By Item 2 of Observation 5.5, it is enough to consider $\mathcal{B}_1$.

For $k > 0$, we define the *k-th type profile* of $\mathcal{B}_1$ to be the set of all types of Parikh vectors $(\boldsymbol{p},\boldsymbol{o})$ with $|\boldsymbol{p}| = |\boldsymbol{o}| = k$ that are induced by $\mathcal{B}_1$; i.e. it is the set $\Upsilon(\mathcal{B}_1,k) = \left\{ \tau_{\mathcal{B}_1}(\mathfrak{P}(\alpha),\mathfrak{P}(\beta)) \;\middle|\; (\alpha,\beta) \in \Sigma_I^k \times \Sigma_O^k \right\}$. Clearly, there is only a finite number of type profiles, as $\Upsilon(\mathcal{B}_1,k) \subseteq \mathbb{B}^{S'\times S'}$, where S' is the state space of $\mathcal{B}_1$. Therefore, as k increases, after some finite K_0, every type profile that is ever attained will have been encountered already. We now place an upper bound on K_0.

Lemma 5.7. *We can effectively compute $K_0 > 0$ such that for every $k > 0$ there exists $k' \leq K_0$ with $\Upsilon(\mathcal{B}_1,k') = \Upsilon(\mathcal{B}_1,k)$.*

Proof. Consider a type τ, and let Ψ_τ be the PA formula constructed as per Lemma 5.4 for the NFA $\mathcal{B}_1$. Observe that for a Parikh vector $(\boldsymbol{p},\boldsymbol{o})$ and for $k > 0$, the expression $|\boldsymbol{p}| = |\boldsymbol{o}| = k$ is PA definable. Indeed, writing $\boldsymbol{p} = (x_1,\ldots,x_{|\Sigma_I|})$ and $\boldsymbol{q} = (y_1,\ldots,y_{|\Sigma_O|})$, the expression is defined by $x_1 + \ldots + x_{|\Sigma_I|} = k \wedge y_1 + \ldots + y_{|\Sigma_O|} = k$.

Let $T \subseteq \mathbb{B}^{S'\times S'}$ be a set of types (i.e., a potential type profile). We define a PA formula $\Theta_T(z)$ over a single free variable z such that $k \models \Theta_T(z)$ iff $\Upsilon(\mathcal{B}_1,k) = T$, as follows.

$$\Theta_T(z) = \left(\forall \boldsymbol{p},\boldsymbol{o}, |\boldsymbol{p}| = |\boldsymbol{o}| = z \to \bigvee_{\tau\in T} \Psi_\tau(\boldsymbol{p},\boldsymbol{o})\right) \wedge \left(\bigwedge_{\tau\in T} \exists \boldsymbol{p},\boldsymbol{o}, |\boldsymbol{p}| = |\boldsymbol{o}| = z \wedge \Psi_\tau(\boldsymbol{p},\boldsymbol{o})\right)$$

Intuitively, $\Theta_T(z)$ states that every Parikh vector $(\boldsymbol{p},\boldsymbol{o})$ with $|\boldsymbol{p}| = |\boldsymbol{o}| = z$ has a type within T, and that all the types in T are attained by some such Parikh vector.

By [FR74, BT76], we can effectively determine for every T whether $\Theta_T(z)$ is satisfiable and, if it is, find a witness M_T such that $M_T \models \Theta_T(z)$. By doing so for every set $T \subseteq \mathbb{B}^{S'\times S'}$, we can set $K_0 = \max\{ M_T \mid \Theta_T(z) \text{ is satisfiable} \}$. Then, for every $k > K_0$ if $\Upsilon(\mathcal{B}_1,k) = T$, then T has already been encountered at $M_T \leq K_0$, as required. □

The purpose of the bound K_0 obtained in Lemma 5.7 is to bound the minimal k for which $\mathcal{T}_1 \prec_{k,\Lambda} \mathcal{T}_2$, or equivalently $L(\texttt{Perm}_k(\mathcal{B}_1)) \subseteq L(\texttt{Perm}_k(\mathcal{B}_2))$ (by Lemma 4.4 and Observation 5.6). This is captured in the following.

Lemma 5.8. *Let $k, k' > 0$ such that $k \neq k'$ and $\Upsilon(\mathcal{B}_1, k') = \Upsilon(\mathcal{B}_1, k)$, then we have $L(\mathtt{Perm}_k(\mathcal{B}_1)) \subseteq L(\mathtt{Perm}_k(\mathcal{B}_2))$ iff $L(\mathtt{Perm}_{k'}(\mathcal{B}_1)) \subseteq L(\mathtt{Perm}_{k'}(\mathcal{B}_2))$.*

Proof. By the symmetry between k and k', it suffices to prove w.l.o.g. that if $L(\mathtt{Perm}_k(\mathcal{B}_1)) \subseteq L(\mathtt{Perm}_k(\mathcal{B}_2))$, then $L(\mathtt{Perm}_{k'}(\mathcal{B}_1)) \subseteq L(\mathtt{Perm}_{k'}(\mathcal{B}_2))$.

Assume the former, and let $w = (x', y') \in L(\mathtt{Perm}_{k'}(\mathcal{B}_1))$, where $(x', y') \in (\Sigma_I^{k'} \times \Sigma_O^{k'})^*$, and we denote $(x', y') = (\alpha'_1, \beta'_1) \cdots (\alpha'_n, \beta'_n)$ with $(\alpha'_j, \beta'_j) \in \Sigma_I^{k'} \times \Sigma_O^{k'}$ for every $1 \leq j \leq n$.

Since $\Upsilon(\mathcal{B}_1, k') = \Upsilon(\mathcal{B}_1, k)$, there is a mapping φ that takes every letter $(\alpha'_j, \beta'_j) \in \Sigma_I^{k'} \times \Sigma_O^{k'}$ in w to a letter $(\alpha_j, \beta_j) \in \Sigma_I^{k} \times \Sigma_O^{k}$ that has same type in $\mathtt{Perm}_k(\mathcal{B}_1)$, so that we can find $(x, y) = (\alpha_1, \beta_1) \cdots (\alpha_n, \beta_n)$ such that for every $1 \leq j \leq n$ we have $\tau_{\mathcal{B}_1}(\mathfrak{P}(\alpha_j), \mathfrak{P}(\beta_j)) = \tau_{\mathcal{B}_1}(\mathfrak{P}(\alpha'_j), \mathfrak{P}(\beta'_j))$.

By the definition of the type of a Parikh vector, we have that

$$\tau_{\mathtt{Perm}_k(\mathcal{B}_1)}(\alpha_j, \beta_j) = \tau_{\mathcal{B}_1}(\mathfrak{P}(\alpha_j), \mathfrak{P}(\beta_j)) = \tau_{\mathcal{B}_1}(\mathfrak{P}(\alpha'_j), \mathfrak{P}(\beta'_j)) = \tau_{\mathtt{Perm}_{k'}(\mathcal{B}_1)}(\alpha'_j, \beta'_j).$$

In particular, since the type of a word is the concatenation (i.e., Boolean matrix product) of its underlying letters, we have that $\tau_{\mathtt{Perm}_k(\mathcal{B}_1)}(x, y) = \tau_{\mathtt{Perm}_{k'}(\mathcal{B}_1)}(x', y')$. Since $(x', y') \in L(\mathtt{Perm}_{k'}(\mathcal{B}_1))$, it follows that also $(x, y) \in L(\mathtt{Perm}_k(\mathcal{B}_1))$. Indeed, $(\tau_{\mathtt{Perm}_{k'}(\mathcal{B}_1)}(x', y'))_{s_0^1, s_f^1} = 1$ where s_0^1 and s_f^1 are an initial state and an accepting state of $\mathtt{Perm}_{k'}(\mathcal{B}_1)$, respectively. But the equality of types implies $\left(\tau_{\mathtt{Perm}_k(\mathcal{B}_1)}(x, y)\right)_{s_0^1, s_f^1} = 1$ as well, so $\mathtt{Perm}_k(\mathcal{B}_1)$ has an accepting run on (x, y).

By our assumption, $L(\mathtt{Perm}_k(\mathcal{B}_1)) \subseteq L(\mathtt{Perm}_k(\mathcal{B}_2))$, so $(x, y) = \varphi(w) \in L(\mathtt{Perm}_k(\mathcal{B}_2))$, or equivalently, $\varphi(w) \in L(\mathtt{Perm}_k(\mathcal{B}_2))$. We now essentially reverse the arguments above, but with $\mathcal{B}_2$ instead of $\mathcal{B}_1$. However, this needs to be done carefully, so that the mapping of letters lands us back at (x', y'), and not a different word. Thus, instead of finding a round equivalent word, we observe that for every $1 \leq j \leq n$, we also have

$$\tau_{\mathtt{Perm}_k(\mathcal{B}_2)}(\alpha_j, \beta_j) = \tau_{\mathcal{B}_2}(\mathfrak{P}(\alpha_j), \mathfrak{P}(\beta_j)) = \tau_{\mathcal{B}_2}(\mathfrak{P}(\alpha'_j), \mathfrak{P}(\beta'_j)) = \tau_{\mathtt{Perm}_{k'}(\mathcal{B}_2)}(\alpha'_j, \beta'_j),$$

This follows from Item 2 in Observation 5.5 and the fact that the permutation closure depends only on the transitions (and not on accepting states, which are the only difference between $\mathcal{B}_1$ and $\mathcal{B}_2$).

Thus, similarly to the arguments above, we have that $(x', y') \in L(\mathtt{Perm}_{k'}(\mathcal{B}_2))$, and the mapping applied is in fact the the inverse map φ^{-1}, where $\varphi^{-1}(\varphi(w)) = w$. We conclude that $L(\mathtt{Perm}_{k'}(\mathcal{B}_1)) \subseteq L(\mathtt{Perm}_{k'}(\mathcal{B}_2))$, as required.

The mapping is illustrated in fig. 5. ☐

$$\begin{tikzcd}
\substack{w \in \mathtt{Perm}_{k'}(\mathcal{B}_1) \\ w = (x_1, y_1)(x_2, y_2) \cdots (x_n, y_n)} \arrow[d, "\varphi"] & \varphi^{-1}(\varphi(w)) = w \in \mathtt{Perm}_{k'}(\mathcal{B}_2) \\
\substack{\varphi(w) \in \mathtt{Perm}_k(\mathcal{B}_1) \\ \varphi(w) = \varphi((x_1, y_1))\, \varphi((x_2, y_2)) \cdots \varphi((x_n, y_n))} \arrow[r, "\subseteq"] & \varphi(w) \in \mathtt{Perm}_k(\mathcal{B}_2) \arrow[u, "\varphi^{-1}"]
\end{tikzcd}$$

Figure 5: A diagram for the proof structure of Lemma 5.8.

Combining Lemmas 5.7 and 5.8, we can effectively compute K_0 such that if it holds that $L(\mathtt{Perm}_k(\mathcal{B}_1)) \subseteq L(\mathtt{Perm}_k(\mathcal{B}_2))$ for some k, then this also holds for some $k < K_0$. Finally, using Lemma 4.4, this concludes the proof of Theorem 5.1. □

Remark 5.9 (Complexity results for Theorem 5.1 and Corollary 5.2)**.** Let n be the number of states in $\mathcal{T}_1 \times \mathcal{T}_2$. Observe that the formula Ψ_τ constructed in Lemma 5.4 comprises a conjunction of $O(n^2)$ PA subformulas, where each subformula is either an existential PA formula of length $O(n)$, or the negation of one. Then, the formula Θ_T in Lemma 5.7 consists of a universal quantification, nesting a disjunction over $|T|$ formulas of the form Ψ_τ, conjuncted with $|T|$ existential quantifications, nesting a single Ψ_τ each. Overall, this amounts to a formula of length $|T| \leq 2^{n^2}$, with alternation depth 3. [3]

Using quantifier elimination [Coo72, Opp78], we can obtain a witness for the satisfiability of Θ_T of size 4-exponential in n^2. Then, finding the overall bound K_0 amounts to $2^{2^{n^2}}$ calls to find such witnesses. Finally, we need K_0 oracle calls to Lemma 4.4 in order to decide existential simulation, and since K_0 may have a 4-exponential size description, this approach yields a whopping 5-EXP algorithm. This approach, however, does not exploit any of the structure of Θ_T.

5.3. Lower Bounds for Existential Round Simulation.

The complexity bounds in Remark 5.9 are naively analyzed, and we leave it for future work to conduct a more in-depth analysis. In this section, we present lower bounds to delimit the complexity gap. Note that there are two relevant lower bounds: one on the complexity of deciding round simulation, and the other on the minimal value of K_0 in Theorem 5.1.

We start with the complexity lower bound, which applies already for round equivalence.

Theorem 5.10. *The problem of deciding, given transducers $\mathcal{T}_1, \mathcal{T}_2$, whether $\mathcal{T}_1 \equiv_{k,\Lambda} \mathcal{T}_2$ for* any *k, is PSPACE-hard, even for Λ of a constant size (given as a 5-state* DFA*).*

Proof sketch. We present a similar reduction to that of Theorem 4.7 from universality of NFAs (see appendix A.2). In order to account for the unknown value of k, we allow padding words with a fresh symbol #, which is essentially ignored by the transducers. □

Next, we show that the minimal value for K_0 can be exponential in the size of the given transducers (in particular, of $\mathcal{T}_2$).

Example 5.11 (Exponential round length)**.** Let $p_1, p_2, \ldots, p_m$ be the first m prime numbers. We define two transducers $\mathcal{T}_1$ and $\mathcal{T}_2$ over input and output alphabet $\mathcal{P} = \{1, \ldots, m\}$, as depicted in fig. 6 for $m = 3$. Intuitively, $\mathcal{T}_1$ reads input $w \in \Lambda = (1 \cdot 2 \cdots m)^*$ and simply outputs w, whereas $\mathcal{T}_2$ works by reading a letter $i \in \mathcal{P}$, and then outputting i for p_i steps (while reading p_i arbitrary letters) before getting ready to read a new letter i.

In order for $\mathcal{T}_2$ to k-round simulate $\mathcal{T}_1$, it must be able to output a permutation of $(1 \cdot 2 \cdots m)^*$. In particular, the number of 1's, 2's, etc. must be equal, so k must divide every prime up to p_m, hence it must be exponential in the size of $\mathcal{T}_2$.

The sum of the number of states in $\mathcal{T}_1$ and $\mathcal{T}_2$ is $1 + m + \sum_{i=1}^m p_i = \mathsf{O}\left(\sum_{i=1}^m p_i\right)$. Set $Q = \prod_{i=1}^m p_i$. It is easily verified that $\mathcal{T}_1 \prec_k \mathcal{T}_2$ holds for $k = m \cdot Q$, which is exponential in the number of states. Indeed, for the round $w = (1 \cdots m)^Q$, we consider the permutation $1^Q \cdots m^Q$, on which the run of $\mathcal{T}_2$ induces the same output.

[3]Alternation depth is usually counted with the outermost quantifier being existential, which is not the case here, hence 3 instead of 2.

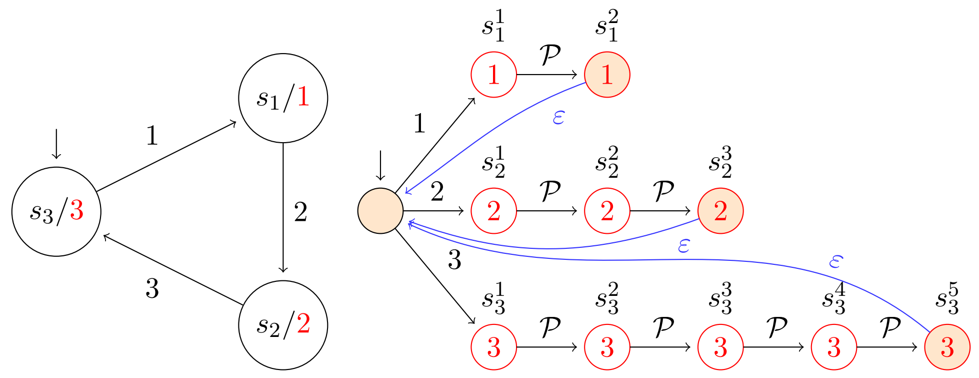


Figure 6: The transducers $\mathcal{T}_1$ (left) and $\mathcal{T}_2$ (right) for $m = 3$ in Example 5.11. The transition $s \xrightarrow{\varepsilon} t$ in $\mathcal{T}_2$ means that the transition function from state s behaves identically as from t.

We now show that this k is minimal. For a word $x \in (1 \cdot 2 \cdots m)^*$ in rounds of k to have round equivalent outputs in $\mathcal{T}_1$ and $\mathcal{T}_2$, there must be some word round equivalent word x' in which every appearance of $i \in \mathcal{P}$ is part of a sequence of appearances of i, of length p_i, except maybe at its end. If $m \mid k$, then there are $\frac{k}{m}$ appearances of each i, so $\frac{k}{m}$ must be divisible by all primes, except maybe one. The latter possibility is falsified when considering the next round. If, however, $m \nmid k$, then in the next round, $1 \in \mathcal{P}$ will have one less appearance than in the first round. This, again, makes impossible the round equivalence of the outputs when considering one additional round.

6. From Process Symmetry to Round Equivalence

As mentioned in section 1, our original motivation for studying round simulation comes from process symmetry. We present process symmetry with an example before introducing the formal model. Recall the Round Robin scheduler from Example 3.3. There, at each time step, the scheduler receives as input the IDs of processes in $\mathcal{P} = \{0, 1, 2\}$ that are making a request, and it responds with the IDs of those that are granted (either a singleton $\{i\}$ or $\emptyset$).

In process symmetry, we consider a setting where the identifiers of the processes may be permuted. This corresponds to the IDs representing, for instance, ports, and the processes not knowing which port they are plugged into. Thus, the input received may be a permutation of the actual identifiers of the processes. Note that a permutation in this case is a bijection over identifiers, not indices as in previous sections. Then, we say that a transducer is *process symmetric* if the outputs are permuted in a way that matches the permutation of identifiers. For example, in the RR scheduler of Example 3.3, the output corresponding to input $\{1, 2\}\{3\}\{3\}$ is $\{1\}\emptyset\{3\}$. However, if we permute the identifiers by swapping processes 1 and 3, we obtain the input $\{3, 2\}\{1\}\{1\}$. Then, the output of RR is $\emptyset\emptyset\emptyset$, demonstrating that RR is not process symmetric. Indeed, the output letters have to be permuted in the same manner as the input for RR to be process symmetric.

In [Alm20], several definitions of process symmetry are studied for probabilistic transducers. In the deterministic case, however, process symmetry is a very strict requirement. In order to overcome this, we allow some flexibility by letting the transducer do local reordering in the word to account for the input permutation. For instance, if we are allowed to rearrange the input $\{3,2\}\{1\}\{1\}$ to $\{1\}\{1\}\{3,2\}$, then the output becomes $\{1\}\emptyset\{3\}$, and once we apply the inverse permutation, this becomes $\{3\}\emptyset\{1\}$. This, in turn, can be again rearranged to obtain the original output $\{1\}\emptyset\{3\}$. In this sense, the scheduler is "locally stable" against permutations of the identifiers of processes.

We now turn to give the formal model. Consider a set of processes $\mathcal{P} = \{1,\dots,m\}$ and $k > 0$. For a permutation π of $\mathcal{P}$ (i.e. a bijection $\pi : \mathcal{P} \to \mathcal{P}$) and a letter $\sigma \in 2^{\mathcal{P}}$, we obtain $\pi(\sigma) = \{\pi(i) : i \in \sigma\} \in 2^{\mathcal{P}}$ by applying π to each process in σ. We lift this to words $x \in (2^{\mathcal{P}})^*$ by applying the permutation letter-wise to obtain $\pi(x)$. We now say that a $2^{\mathcal{P}}/2^{\mathcal{P}}$ transducer $\mathcal{T} = \langle 2^{\mathcal{P}}, 2^{\mathcal{P}}, Q, q_0, \delta, \boldsymbol{\ell} \rangle$ is *k-round symmetric* if for every permutation π of $\mathcal{P}$ and for every k-round word $x \in (2^{\mathcal{P}})^*$ there exists $x' \in (2^{\mathcal{P}})^*$ such that $\pi(x) \asymp_k x'$ and $\pi(\mathcal{T}(x)) \asymp_k \mathcal{T}(x')$. We say that $\mathcal{T}$ is k-round symmetric w.r.t. π if the above holds for a fixed permutation π.

Example 6.1. Consider the RR scheduler for n processes (cf. Example 3.3), and let $\mathcal{T}$ be a transducer for it. As discussed above, $\mathcal{T}$ is not process symmetric. Intuitively, however, RR is symmetric in the sense that all processes are "treated equally" within each round. We now show that round symmetry captures this property.

Consider for example the input word $x = \{0,2\}\{1\}\{2\}$ over $\mathcal{P} = \{0,1,2\}$, and let $\pi = (0\ 1)$ be a permutation swapping processes 0 and 1. We have that $\pi(x) = \{1,2\}\{0\}\{2\}$. Observe that $\mathcal{T}(x) = \{0\},\{1\},\{2\}$, meaning all processes are granted. We can now choose $x' = \{0\}\{1,2\}\{2\}$ so that $x' \asymp_3 x$, and we have that $\mathcal{T}(x') = \{0\},\{1\},\{2\}$. and in particular $\mathcal{T}(x') \asymp_3 \pi(\mathcal{T}(x))$, since $\pi(\mathcal{T}(x)) = \{1\},\{0\},\{2\}$.

In general, consider a permutation $\pi \in \mathcal{S}_n$ applied to the signals. We can then preserve the behaviour of the system (i.e. the identifiers of the process that receive grants) by reordering the requests. Indeed, given input x, consider the i-th round $b_1 b_2 \cdots b_n$ of $\pi(x)$. We obtain x' by setting the i-th round to $b_{\pi^{-1}(1)} b_{\pi^{-1}(2)} \cdots b_{\pi^{-1}(n)}$. Then, it holds that $\mathcal{T}(x) = \pi^{-1}(\mathcal{T}(x'))$ or equivalently, $\pi(\mathcal{T}(x)) = \mathcal{T}(x')$, so RR is n-round symmetric.

Example 6.1 shows that RR exhibits round symmetry w.r.t. all permutations. In the general sense, round symmetry might hold w.r.t. some permutations but not others, as is the case in the following.

Example 6.2. Fix $\mathcal{P} = \{0,1,2\}$ and let $\mathcal{T}$ be the $2^{\mathcal{P}}/2^{\mathcal{P}}$ transducer illustrated in fig. 7. It is not difficult to see that $\mathcal{T}$ satisfies 2-round symmetry w.r.t. $\pi = (0\ 1)$ but not w.r.t. e.g. $(0\ 2)$.

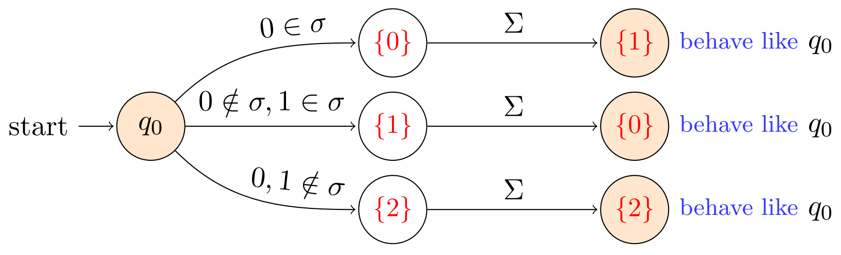


Figure 7: Transducer $\mathcal{T}$ satisfying round symmetry w.r.t. $\pi = (0\ 1)$ but not $(0\ 2)$.

The central decision problems in round symmetry are akin to those of round simulation: in *fixed round symmetry* we are given $\mathcal{T}$ and k and we ask whether $\mathcal{T}$ is k-round symmetric, and in *existential round symmetry* we ask whether there exists $k > 0$ for which this holds. Observe that for round symmetry we have $\Lambda = (2^{\mathcal{P}})^*$, and is therefore ignored in the following.

From round symmetry to round simulation. As we now show, round symmetry can be cast to the setting of round simulation. We start with the case where the permutation π is given.

Consider a transducer $\mathcal{T}$, we obtain from $\mathcal{T}$ a new transducer $\mathcal{T}^\pi$ by applying the permutation π to the actions and labels. Formally, $\mathcal{T}^\pi = \langle 2^{\mathcal{P}}, 2^{\mathcal{P}}, Q, q_0, \delta^\pi, \boldsymbol{\ell}^\pi \rangle$ where $\delta^\pi(q,\sigma) = \delta(q, \pi^{-1}(\sigma))$ and $\boldsymbol{\ell}^\pi(q) = \pi(\boldsymbol{\ell}(q))$. It is easy to verify that for every $x \in (2^{\mathcal{P}})^*$ we have $\mathcal{T}^\pi(x) = \pi(\mathcal{T}(\pi^{-1}(x)))$. Figure 8 shows the transducer $\mathcal{T}^\pi$ that corresponds to $\mathcal{T}$ of Example 6.2 for $\pi = (0\ 1)$.

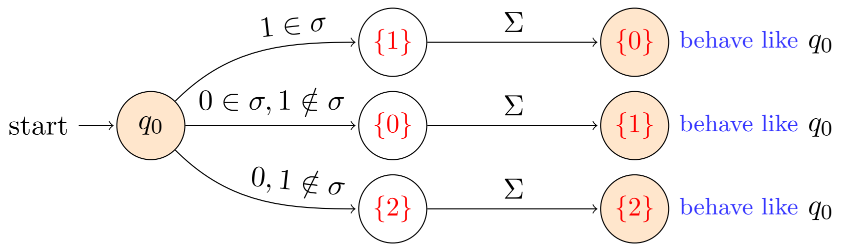


Figure 8: Transducer $\mathcal{T}^\pi$ for the $\mathcal{T}$ in Example 6.2 and $\pi = (0\ 1)$.

Once we have $\mathcal{T}^\pi$, round symmetry can be expressed as round simulation, so we can use the tools developed in sections 4 and 5 to solve the problems at hand.

Lemma 6.3. *For a permutation π and $k > 0$, $\mathcal{T}$ is k-round symmetric w.r.t. π iff $\mathcal{T}^\pi \prec_k \mathcal{T}$.*

Proof. By definition, we have that $\mathcal{T}^\pi \prec_k \mathcal{T}$ iff for every $x \in (2^{\mathcal{P}})^*$ there exists $x' \asymp x$ such that $\mathcal{T}^\pi(x) \asymp \mathcal{T}(x')$. We show that this is equivalent to the definition of round symmetry.

For the first direction, assume $\mathcal{T}$ is k-round symmetric w.r.t. π, and let $x \in (2^{\mathcal{P}})^*$. Applying the definition of k-round symmetry to $y = \pi^{-1}(x)$, there exists $x' \asymp \pi(y)$ such that $\pi(\mathcal{T}(y)) \asymp \mathcal{T}(x')$. Since $\pi(y) = x$ we get that $x' \asymp x$ and $\pi(\mathcal{T}(\pi^{-1}(x))) \asymp \mathcal{T}(x')$. By the above, $\mathcal{T}^\pi(x) = \pi(\mathcal{T}(\pi^{-1}(x)))$, so we have $\mathcal{T}^\pi(x) \asymp \mathcal{T}(x')$.

For the second direction, assume $\mathcal{T}^\pi \prec_k \mathcal{T}$, and let $x \in (2^{\mathcal{P}})^*$. Applying the definition of round simulation to $z = \pi(x)$, there exists $x' \asymp z$ such that $\mathcal{T}^\pi(z) \asymp \mathcal{T}(x')$. Thus, $\pi(\mathcal{T}(\pi^{-1}(z))) \asymp \mathcal{T}(x')$, but $\pi^{-1}(z) = x$, so we get $\pi(\mathcal{T}(x)) \asymp \mathcal{T}(x')$, and we are done. □

Closure under composition. Lemma 6.3 enables us to naively solve fixed round symmetry by checking against all permutations. We show, however, that the definition above is closed under composition of permutations, allowing us to establish round symmetry by checking only two permutations, forming a generating set of $\mathcal{S}_n$.

Lemma 6.4. *Consider two permutations π, χ. If $\mathcal{T}^\pi \prec_k \mathcal{T}$ and $\mathcal{T}^\chi \prec_k \mathcal{T}$ then $\mathcal{T}^{\pi\circ\chi} \prec_k \mathcal{T}$.*

Proof. Using the first definition of round symmetry, let $x \in (2^{\mathcal{P}})^*$, then there exists $x' \asymp_k \pi(x)$ such that $\mathcal{T}(x') \asymp_k \pi(\mathcal{T}(x))$. Moreover, there exists $x'' \asymp_k \chi(x') \asymp_k \chi(\pi(x))$ such that $\mathcal{T}(x'') \asymp_k \chi(\mathcal{T}(x')) \asymp_k \chi(\pi(\mathcal{T}(x)))$, and we are done. □

Recall that the group of all permutations of $\mathcal{P} = \{1, \ldots, m\}$ is generated by two permutations: the transposition $(1\ 2)$ and the cycle $(1\ 2\ \cdots\ m)$ [C+99]. By Lemma 6.4 it is sufficient to check symmetry for these two generators in order to obtain symmetry for every permutation. Note that for the existential variant of the problem, even if every permutation requires a different k, by taking the product of the different values we conclude that there is a uniform k for all permutations. We thus have the following.

Theorem 6.5. *Both fixed and existential round symmetry are decidable. Moreover, fixed round symmetry is in PSPACE.*

Finally, the reader may notice that our definition of round symmetry w.r.t. π is not symmetric, as was the case with round simulation compared to round equivalence. However, when we consider round symmetry w.r.t. to all permutations, the definition becomes inherently symmetric, as a consequence of Lemma 6.4.

Lemma 6.6. *In the notations above, if $\mathcal{T}^\pi \prec_k \mathcal{T}$ then $\mathcal{T} \prec_k \mathcal{T}^\pi$.*

Proof. Recall that for every permutation π we have $\pi^{m!} = \mathtt{id}$, where $\mathtt{id}$ is the identity permutation. In particular, $\pi^{m!-1} = \pi^{-1}$.

By Lemma 6.4, we now have that if $\mathcal{T}^\pi \prec_k \mathcal{T}$, then $\mathcal{T}^{\pi^{m!-1}} \prec_k \mathcal{T}$, so $\mathcal{T}^{\pi^{-1}} \prec_k \mathcal{T}$. Applying π to both sides gives us $\mathcal{T} \prec_k \mathcal{T}^\pi$. □

Thus, for symmetry, the notions of round simulation and round equivalence coincide.

7. The Simulation Mapping

The definition of round simulation in section 3 has an existential flavour: given input x we consider the existence of a word x' that satisfies the requirement of round simulation. In some cases it may be desirable to compute an x' that "witnesses" the simulation of x.

For example, recall the monitor of Example 1.1 modelled by a transducer $\mathcal{T}_1$. Recall that we presented a simpler transducer $\mathcal{T}_2$ that round-simulates $\mathcal{T}_1$. This allowed us then to verify e.g., the property "if there is no `error`, then Process 3 works at least once every 20 steps" against the much smaller $\mathcal{T}_2$. When a designer wishes to gain understanding as to why the verification on $\mathcal{T}_2$ is sound, they may want to see how input sequences/output sequences for $\mathcal{T}_1$ are translated to $\mathcal{T}_2$. In this example, the transformation is simple, and consists of ordering the process by their id.

Clearly one can compute x' from x by simply trying all permutations of x and finding a successful one. This, however, is expensive, and raises the question of whether we can output x' using a finite-state transducer. Unfortunately, we show in the following that computing x' cannot be done locally, in the sense that arbitrary lookahead is needed.

Consider two transducers $\mathcal{T}_1$ and $\mathcal{T}_2$ such that $\mathcal{T}_1 \prec_k \mathcal{T}_2$, and an input word $x \in \left(\Sigma_I^k\right)^*$. This means, by definition, that there is a way to permute the rounds in x to obtain a word x' such that $\mathcal{T}_2(x')$ is a permutation of $\mathcal{T}_1(x)$. A *simulation mapping*[4] *between* $\mathcal{T}_1$ *and* $\mathcal{T}_2$ is a function $\psi_{\mathcal{T}_1,\mathcal{T}_2} : \Sigma^* \to \Sigma^*$ such that for every $x \in \Sigma^{kR}$ we have that $x' = \psi_{\mathcal{T}_1,\mathcal{T}_2}(x)$ satisfies $x' \asymp_k x$ and $\mathcal{T}_1(x) \asymp_k \mathcal{T}_2(x')$ (we omit the subscripts when the transducers are clear from context).

We start by showing that the simulation mapping is not a morphism, in the sense that it cannot act on each round separately.

Example 7.1. Consider the transducers $\mathcal{T}_1$ and $\mathcal{T}_2$ depicted in fig. 9, with input and output alphabets $\Sigma_I = \{a, b\}$ and $\Sigma_O = \{0, 1\}$ and round length 2. $\mathcal{T}_1$ expects to see either ab or ba

[4]We omit Λ for brevity. However, it can easily be incorporated.

in every 2-round, outputting 00 in both cases, and otherwise outputs 01 in that round. $\mathcal{T}_2$ expects the first round to be ab and the second to be ba, otherwise outputs 01 in the round not meeting expectations; and beginning from the third round, it behaves like $\mathcal{T}_1$. We have that $\mathcal{T}_1 \prec_2 \mathcal{T}_2$ by a permutation that corrects the order of the letters in the first two rounds of the input. Moreover, we have $\psi(ab) = \psi(ba) = ab$ whereas $\psi(abba) = abba \neq \psi(ab) \cdot \psi(ba)$.

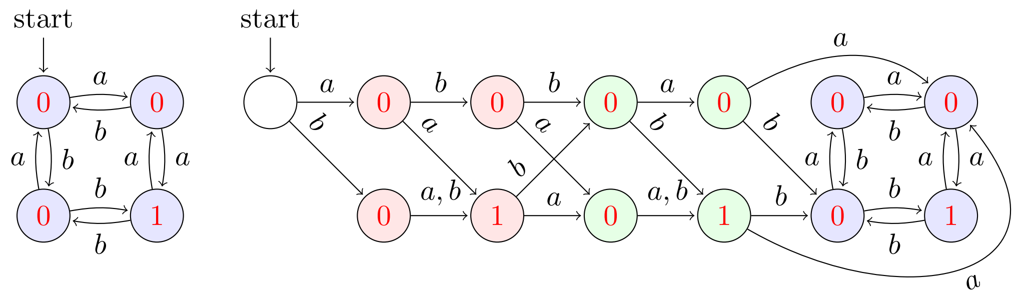


Figure 9: The transducers $\mathcal{T}_1$ (left) and $\mathcal{T}_2$ (right) in Example 7.1. The states of $\mathcal{T}_2$ in red, green and blue manage the first, second and later rounds, respectively.

Next, we show that in fact the simulation mapping cannot be described by any fixed lookahead machine.

Example 7.2. Set $\Lambda = L[ab \cdot (cc)^* \cdot (ab + ba)]$ and $k = 2$, and let $\mathcal{T}_1$ and $\mathcal{T}_2$ be the transducers in fig. 10, satisfying $\mathcal{T}_1 \prec_{k,\Lambda} \mathcal{T}_2$. Denote the simulation mapping by $\psi^* : (\Sigma_I^k)^* \to (\Sigma_I^k)^*$.

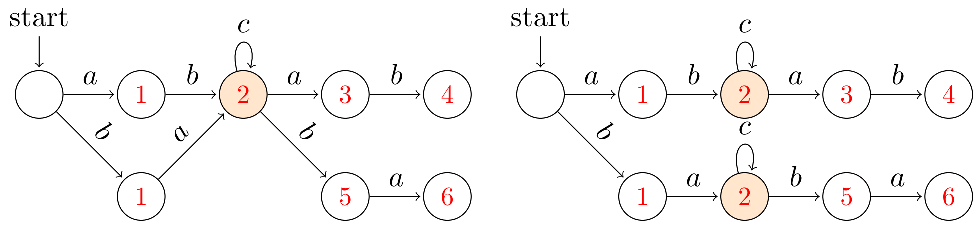


Figure 10: The transducers $\mathcal{T}_1$ (left) and $\mathcal{T}_2$ (right) in Example 7.2.

We claim that for any r, there is no lookahead machine that defines a function $\psi_r : (\Sigma_I^{rk})^* \to (\Sigma_I^{rk})^*$ such that $\psi^*(x) = \psi(x)$ for all input words x.

Indeed, let $r \in \mathbb{N}$, and assume by way of contradiction that such ψ_r exists. Now consider the input word $x = ab \cdot c^{rk-2}$. $\psi_r(x)$ must start with either ab or ba. Without loss of generality, assume the former, and consider the input word $x' := x \cdot ba \cdot c^{rk-2}$. Since ψ_r works on r rounds each time, the first r rounds are fixed when it reads the $(r+1)$-th round. Moreover, since $\psi_r(x')$ must induce a valid path in $\mathcal{T}_2$, the only option for the $(r+1)$-th round of $\psi_r(x')$ is ab. Hence, the output of $\mathcal{T}_1$ on x' is different from the output of $\mathcal{T}_2$ on $\psi(x')$, and we have a contradiction.

Example 7.2 essentially shows that it is generally impossible to determine the output of the first round without knowing the entire input. In section 9 we discuss possible models that may be able to capture it, and are weaker than general Turing machines.

8. Additional Notions of Symmetry and Simulation

Recall that under our definition from section 2, we have that $x \asymp_k y$ if every k-round of x can be permuted to a k-round of y. This permutation, however, can vary between rounds. In some settings, we would want the rounds to be transformed uniformly, with the same permutation. To this end, we introduce below the notion of *uniform round simulation*. In addition, if the underlying alphabet consists of set of signals, as in the setting of section 6, we can also consider simulation where one is allowed to permute the index of each signal, instead of entire letters. To capture this notion, we introduce *signal-wise simulation*. Finally, recall that simulation is defined by permutation of both the input and output letters. Given the new definitions, one can consider simulations where the inputs and outputs are not similarly permuted, e.g., the inputs can be permuted arbitrarily, but the outputs need to be permuted uniformly. In the following, we discuss these notions and their interrelations.

For brevity, we omit Λ from this discussion, as it is an orthogonal restriction and can be easily incorporated to the setting.

8.1. Variations of Round Symmetry and Round Simulation.

We start by formally defining new notions of simulation. For this section, we consider $2^{\mathcal{P}}/2^{\mathcal{P}}$ transducers[5] for $\mathcal{P} = \{1, \ldots, n\}$.

Consider two words $x, y \in (2^{\mathcal{P}})^*$ of length kR. We say that x, y are *uniformly round equivalent* and denote by $x \asymp_k^{\mathsf{u}} y$ if $x \asymp_k y$ and there exists a single permutation $\tau \in \mathcal{S}_k$ which transforms the rounds of x to those of y. We say that x, y are *signal-wise round equivalent*, denoted $x \asymp_k^{\mathsf{s}} y$, if for each k-round, x and y have the same number of occurrences of each signal. More precisely, for each signal $p \in \mathcal{P}$ and round $0 \le i < R$, we have $|\{j : p \in x_{ik+j}, 1 \le j \le k\}| = |\{j : p \in y_{ik+j}, 1 \le j \le k\}|$. For clarity, we explicitly denote our original definition of round equivalence by $x \asymp_k^{\ell} y$, where ℓ stands for "letter" round equivalence. We refer to the three types of round equivalence as *modes*.

The new definitions lift to simulation of transducers, by specifying which type of round equivalence is used on the inputs and outputs. We thus obtain 9 definitions of simulation, as follows. Consider transducers $\mathcal{T}_1, \mathcal{T}_2$, and let $\mu, \mu' \in \{s, \ell, u\}$ be modes of round equivalence. We write $\mathcal{T}_1 \prec_k^{\mu,\mu'} \mathcal{T}_2$ if for every input x there exists $x' \asymp_k^{\mu} x$ such that $\mathcal{T}_1(x) \asymp_k^{\mu'} \mathcal{T}_2(x')$. This definition is in turn lifted to symmetry, as per section 6, by replacing $\mathcal{T}_2$ with $\mathcal{T}^{\pi}$ for a permutation π of the signals.

Example 8.1 (Round Robin is uniform symmetric)**.** Consider the RR scheduler for n processes, shown to be n-round symmetric in Example 6.1. Recall that in the proof of its symmetry when the permutation π was applied to the signals, we had to change the order of handling the requests such that it matched the new order of received requests: given input x, for the i-th round $b_1 b_2 \cdots b_n$ of $\pi(x)$ (the input under permutation π) we set the corresponding round in x' to $b_{\pi^{-1}(1)} b_{\pi^{-1}(2)} \cdots b_{\pi^{-1}(n)}$. Since the same permutation π was applied for all rounds of the input x, the permutation by which the rounds of x' were obtained was identical for all rounds. It follows that RR exhibits uniform round symmetry, i.e., $\mathcal{T} \prec_k^{u,u} \mathcal{T}^{\pi}$.

The modes of equivalence can be compared by their strictness, with uniform equivalence implying letter-wise, which in turn implies signal-wise. This can be lifted to round equivalence, yielding a partial order on the strictness of the various definitions, as depicted in fig. 11.

[5]the choice of $2^{\mathcal{P}}$ as both the input and output alphabet is arbitrary.

$$\begin{tikzcd}
 & & \langle s,s,k\rangle & & \\
 & \langle s,\ell,k\rangle \arrow[ur] & & \langle \ell,s,k\rangle \arrow[ul] & \\
\langle s,u,k\rangle \arrow[ur] & & \langle \ell,\ell,k\rangle \arrow[ul] \arrow[ur] & & \langle u,s,k\rangle \arrow[ul] \\
 & \langle \ell,u,k\rangle \arrow[ul] \arrow[ur] & & \langle u,\ell,k\rangle \arrow[ul] \arrow[ur] & \\
 & & \langle u,u,k\rangle \arrow[ul] \arrow[ur] & &
\end{tikzcd}$$

Figure 11: A Hasse diagram for the partial order on the strictness of the definitions, where $\alpha \to \beta$ means α implies β.

In the remainder of the section, we give some examples separating some of the definitions, thus showing the order in fig. 11 is strict. Similar examples can be constructed for separating the rest of the definitions.

Example 8.2 (Symbol-wise symmetry does not imply letter-wise symmetry)**.** We warm up by showing that $\langle \ell,\ell,k\rangle$ is more strict than $\langle s,s,k\rangle$ (we will later reuse this example to establish finer strictness results). Set $\pi = (0\ 1)$ and let $k \in \mathbb{N}$ and $m \geq 3$. We construct a transducer that is symbol-wise k-round symmetric, but not letter-wise k'-round symmetric for any k'.

Consider the $2^{\mathcal{P}}/2^{\mathcal{P}}$ transducer $\mathcal{T} = \langle 2^{\mathcal{P}}, 2^{\mathcal{P}}, S, s_0, \delta, \boldsymbol{\ell}\rangle$ depicted in fig. 12, where $\mathcal{P} = [m] = \{0, \cdots, m-1\}$.

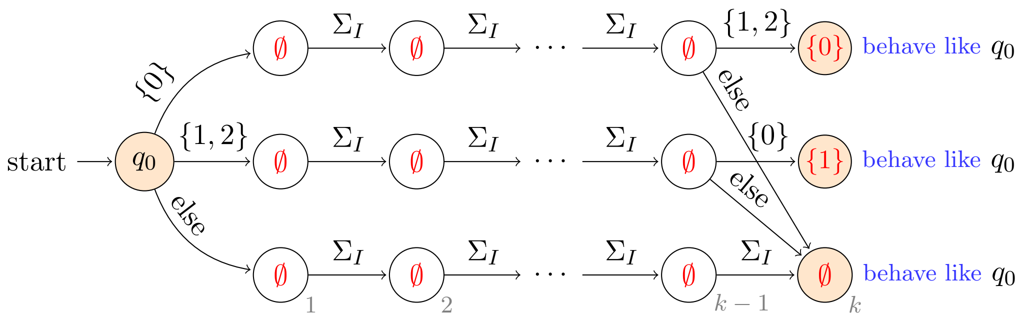


Figure 12: $\mathcal{T}$ exhibits symbol, but not letter-wise, round symmetry (see Example 8.2).

Observe that every round starts at q_0. There are three possible forms for the output of each round depending on the input, as summarized in table 3.

We first show that $\mathcal{T}$ is symbol-wise round symmetric. Let x be an input word. Similarly to section 6, $\pi(x)$ is the word obtained from x by permuting every signal according to π. If x is of one of the first two forms in table 3, then by moving the signal $2 \in \mathcal{P}$ (fixed in π) between the first and last letters, we get $x' \asymp^{\mathsf{s}} \pi(x)$ such that $T(x') \asymp^{\mathsf{s}} \pi(T(x))$, as desired. Now assume x is of some other form, having the output $\emptyset^k$. If $2 \in \mathcal{P}$ appears in both the first

Table 3: The inputs and their corresponding outputs in $\mathcal{T}$ of Example 8.2.

Input	Output
$\{0\}\sigma_2 \cdots \sigma_{k-1}\{1,2\}$	$\emptyset^{k-1}\{0\}$
$\{1,2\}\sigma_2 \cdots \sigma_{k-1}\{0\}$	$\emptyset^{k-1}\{1\}$
else	$\emptyset^k$

and last letters, or it appears in neither, then set $x' = \pi(x)$; otherwise, move the signal 2 to the other letter, and the output will remain $\emptyset^k$. Thus, $\mathcal{T}$ is symbol-wise round symmetric.

On the other hand, $\mathcal{T}$ is not letter-wise k'-round symmetric for any $k' > 0$. To see this, take the input $x = \{0\}^{k-1} \cdot \{1,2\} \cdot \emptyset^{k'k-k}$. We have $|x| = k'k$ which is divisible by k', $\mathcal{T}(x) = \emptyset^{k-1} \cdot \{0\} \cdot \emptyset^{k'k-k}$. It holds that $\pi(x) = \{1\}^{k-1} \cdot \{0,2\} \cdot \emptyset^{k'k-k}$, which contains neither the letter $\{0\}$ nor $\{1,2\}$. Thus, regardless of how we permute $\pi(x)$ to obtain x', the output of any $x' \asymp^{\ell} \pi(x)$ is always $\emptyset^{k'k}$, which is not a permutation of $\mathcal{T}(x)$.

Example 8.3 (Showing $\langle \mathsf{s}, \mathsf{s}, k\rangle \lneq \langle \ell, \mathsf{s}, k\rangle$)**.** Let $\mathcal{T}$ be the transducer from Example 8.2, and consider the transducer $\mathcal{T}^\pi$ obtained from $\mathcal{T}$ by permuting both the input and the output by $\pi = (0\ 1)$ as in section 6. We have shown that $\mathcal{T}$ is symbol-wise round symmetric. By a reasoning analogous to the transition from symmetry to simulation as per section 6, this gives $\mathcal{T} \prec_k^{\mathsf{s},\mathsf{s}} \mathcal{T}^\pi$. However, it does not hold that $\mathcal{T} \prec_k^{\ell,\mathsf{s}} \mathcal{T}^\pi$: for the input $x := \{0\}\sigma_2 \cdots \sigma_{k-1}\{1,2\}$ having output $y := \emptyset^{k-1}\{0\}$ (cf. table 3), any permutation $x' \asymp_k^{\ell} x$ will lead to an output of $\emptyset^k \not\asymp_k^{\mathsf{s}} y$. Thus $\mathcal{T} \not\prec_k^{\ell,\mathsf{s}} \mathcal{T}^\pi$ (and in particular, $\mathcal{T} \not\prec_k^{\ell,\ell} \mathcal{T}^\pi$ so $\mathcal{T}$ is not letter-wise symmetric). In the general sense, we conclude that $\mathcal{T}_1 \prec_k^{\mathsf{s},\mathsf{s}} \mathcal{T}_2$ does not imply $\mathcal{T}_1 \prec_k^{\ell,\mathsf{s}} \mathcal{T}_2$.

Example 8.4 (Showing $\langle \mathsf{s}, \mathsf{s}, k\rangle \lneq \langle \mathsf{s}, \ell, k\rangle$)**.** Consider the transducer $\mathcal{T}$ in fig. 13, whose round-by-round behaviour can once more be summarized in a table (see table 4). $\mathcal{T}$ is symbol-wise round symmetric: for an input x, choose $x' = \pi(x)$. It is not difficult to show that $\mathcal{T}(x') \asymp_k^{\mathsf{s}} \pi(\mathcal{T}(x))$ by considering the possible forms of x according to table 4. To see that $\mathcal{T} \not\prec_k^{s,\ell} \mathcal{T}^\pi$, consider the word $x = \{0\}\emptyset\emptyset$. The output of $\mathcal{T}$ on x is $\{0\}\emptyset\{2\}$. Any round equivalent word x' of x either starts with $\{1\}$ or $\emptyset$, the respective outputs being either $\{1,2\}\emptyset\emptyset$ or $\emptyset^3$. In all cases, we have $T(x') \not\asymp_k^{\ell} \mathcal{T}^\pi(x)$.

Table 4: The inputs and their corresponding outputs in $\mathcal{T}$ of Example 8.4.

Input	Output
$\{0\}(2 \notin \sigma)\,\sigma$	$\{0\}\emptyset\{2\}$
$\{0\}(2 \in \sigma)\,\sigma$	$\{0\}\{2\}\emptyset$
$\{1,2\}(2 \in \sigma)\,\sigma$	$\{1\}\emptyset\{2\}$
$\{1,2\}(2 \notin \sigma)\,\sigma$	$\{1\}\{2\}\emptyset$
$\{0,2\}\sigma\sigma$	$\{0,2\}\emptyset\emptyset$
$\{1\}\sigma\sigma$	$\{1,2\}\emptyset\emptyset$
else	$\emptyset\emptyset\emptyset$

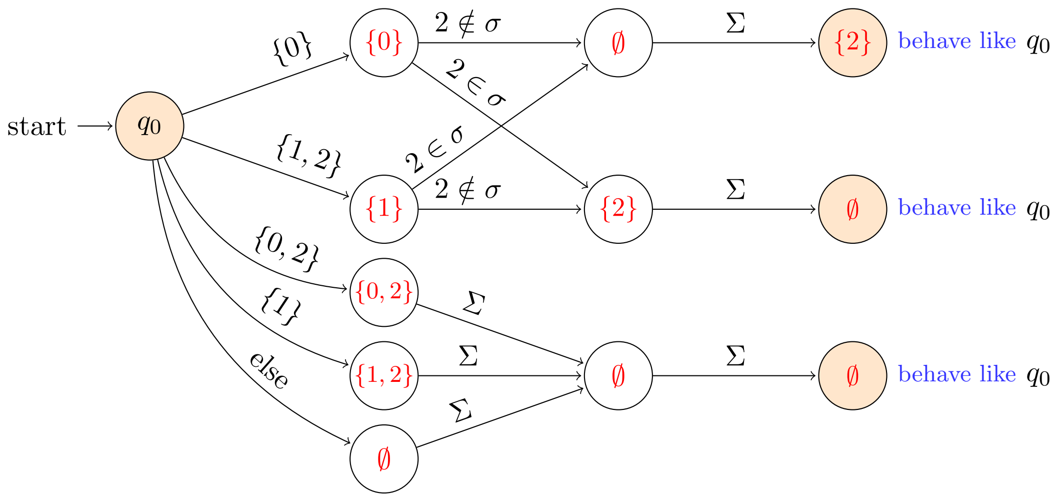


Figure 13: The transducer $\mathcal{T}$ for Example 8.4. The transitions $i \in \sigma$ and $i \notin \sigma$ mean all letters from Σ_I that, respectively, contain or do not contain i.

The transducers used in Examples 8.3 and 8.4 have established two gaps from fig. 11. In fact, these same transducers can be used to establish the remaining two dual gaps as well, as follows. The transducer $\mathcal{T}$ in Example 8.3 satisfies $\langle \mathtt{s}, \ell, k\rangle$-round simulation with its corresponding $\mathcal{T}^\pi$; indeed, observe that the output labels are either singleton sets or empty sets, so that a signal permutation of the output is equivalent to permuting the letters. The transducer $\mathcal{T}$ in Example 8.4 satisfies $\langle \ell, \mathtt{s}, k\rangle$-round simulation with its corresponding $\mathcal{T}^\pi$, which is inferred from the choice of $x' = \pi(x)$, satisfying in particular $x' \asymp^\ell \pi(x)$. However, neither of the two satisfy $\langle \ell, \ell, k\rangle$-round simulation, since they are not symbol-wise round symmetric. This completes the proof of strictness of top diamond in fig. 11. In appendix B we provide constructions to complete some of the remaining strictness results.

Finally, Example B.1 presents a pair of transducers $\mathcal{T}_1$ and $\mathcal{T}_2$ such that $\mathcal{T}_1 \prec_2^{\ell,\mathtt{s}} \mathcal{T}_2$ and $\mathcal{T}_1 \prec_2^{\mathtt{s},\ell} \mathcal{T}_2$, but $\mathcal{T}_1 \not\prec_2^{\ell,\ell} \mathcal{T}_2$. This proves that although $\langle \ell, \ell, k\rangle$-round simulation implies both $\langle \mathtt{s}, \ell, k\rangle$ and $\langle \ell, \mathtt{s}, k\rangle$-round simulation, the converse does not hold.

8.2. **Deciding Round Simulation.**

We briefly discuss the decidability of round simulation for the new notions. We start by considering $\langle \mathtt{s}, \mathtt{s}, k\rangle$-round simulation, where the following arguments also apply when replacing one of the $\mathtt{s}$ with ℓ. The main idea is to tweak the definitions of sections 4 and 5, and specifically the permutation-closure NFA, to look at permutations of the signals, not just the letters. To this end, we simply modify the notion of *Parikh image* over an alphabet $2^{\mathcal{P}}$ to be with respect to $\mathcal{P}$. That is, for $x \in (2^{\mathcal{P}})^*$, let $\mathfrak{P}(x) \in \mathbb{N}^{\mathcal{P}}$ be the vector counting the number of occurrences of each signal $p \in \mathcal{P}$ in the letters of x.

Under this definition, the analysis of sections 4 and 5 follows without any changes. Indeed, the crucial property that is needed for these arguments is that the permutation-closure NFA is indeed closed under permutation, which clearly holds also for the new definition. In particular, the proof of Lemma 4.4 hold, from which the rest of the analysis follows. Thus, adding $\mathtt{s}$ to the model retains the decidability and complexity of both fixed round simulation and existential round simulation.

In contrast, uniform round simulation is conceptually different: the constraint on the permutations of each round is now global for the word. That is, we need a single permutation to be used in all rounds. This means that the techniques of sections 4 and 5 no longer apply. Moreover, uniform round simulation is *not* invariant to (letter or signal) round permutations. Indeed, clearly there are words $x \asymp_k^\ell x'$ and y such that $x \asymp_k^{\mathrm{u}} y$ but $x' \not\asymp_k^{\mathrm{u}} y$.

For fixed round simulation, enforcing the global condition is not too difficult, as we now show.

Theorem 8.5. *Given transducers $\mathcal{T}_1, \mathcal{T}_2$ and $k > 0$ in unary, the problem of deciding whether $\mathcal{T}_1 \prec_{k,}^{u,u} \mathcal{T}_2$ is in PSPACE.*

Proof. Recall that $\mathcal{T}_1 \prec_k^{u,u} \mathcal{T}_2$ iff for every x there exist permutations π, τ such that $\pi(x) = y$ (where $\pi(x)$ is the word obtained by applying π to each k-round of x) and $\tau(\mathcal{T}_1(x)) = \mathcal{T}_2(y)$.

Let $\mathcal{D}_1^k$ and $\mathcal{D}_2^k$ be the trace DFAs of $\mathcal{T}_1$ and $\mathcal{T}_2$ as per section 4, where we modify them to read the alphabet $\Sigma_I^k \times \Sigma_O^{)}$ (in this setting $\Sigma_I = \Sigma_O = 2^{\mathcal{P}}$). Next, for permutations π, τ as above, define $\mathcal{A}_1^{\pi,\tau}$ to be the DFA obtained from $\mathcal{D}_1^k$ by, intuitively, applying π, τ to $\Sigma_I^k \times \Sigma_O^k$. Formally, let $\delta : Q \times (\Sigma_I^k \times \Sigma_O^k) \to Q$ be the transition function of $\mathcal{D}_1^k$, then the transition function of $\mathcal{A}_1^{\pi,\tau}$ is given by $\mu(q, (\alpha, \beta)) = \delta(q, (\pi(\alpha), \tau(\beta)))$. We now obtain an NFA $\mathcal{A}$ by taking the union of $\mathcal{A}_1^{\pi,\tau}$ over all permutations π, τ. It is easy to see that $\mathcal{T}_1 \prec_k^{u,u} \mathcal{T}_2$ iff $L(\mathcal{A}) \subseteq \mathcal{D}_2^k$.

Since the size of $\mathcal{A}$ is single-exponential in that of $\mathcal{D}_1^k$, but can be construction on-the-fly, the latter containment can be decided in PSPACE. □

Theorem 8.5 can be easily combined with the remaining notions to obtain the decidabilty of all nine definitions of fixed round simulation.

Remark 8.6. Unfortunately, the construction in the proof of Theorem 8.5 significantly modifies the state space of $\mathcal{D}_1^k$. This is in contrast to the construction in Lemma 4.4, which only modifies the transition function.

In particular, it is not clear if the construction can be symbolically defined via e.g., Presburger Arithmetic (or some other decidable logic) in order to extend decidability to the existential-bound setting. We therefore leave the latter as an open problem.

9. Conclusion and Open Questions

In this work, we introduced round simulation and provided decision procedures and lower bounds (some with remaining gaps) for the related algorithmic problems. Our framework can be viewed as a notion of "approximate simulation", by which we can significantly reduce the state space for verification, at the cost of invariance to permutations.

Round simulation, and in particular its application to round symmetry, is only an instantiation of a more general framework of symmetry, by which we measure the stability of transducers under local changes to the input. In particular, there is place for additional notions of symmetry and simulation to be studied, and the existing ones extended. Some such variants were presented and discussed in section 8.1, but others, e.g., sliding-window symmetry, or the setting of infinite words may also be of interest in future works.

A few gaps have remained open in this work. Most notably are tightening the complexity gap of existential simulation Remark 5.9, and implementing the simulation mapping from section 7 using a simpler computational model than Turing machines. Some possible candidates for the latter are streaming-string transducers and bi-machines [MP19].

References

[AA22] A. Abu Nassar and S. Almagor. Simulation by rounds of letter-to-letter transducers. In Florin Manea and Alex Simpson, editors, *30th EACSL Annual Conference on Computer Science Logic, CSL 2022, February 14-19, 2022, Göttingen, Germany (Virtual Conference)*, volume 216 of *LIPIcs*, pages 3:1–3:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022.

[Alm20] S. Almagor. Process symmetry in probabilistic transducers. In *40th International Conference on Foundation of Software Technology and Theoretical Computer Science, FSTTCS 2020*, 2020.

[BS73] J. A. Brzozowski and I. Simon. Characterizations of locally testable events. *Discrete Mathematics*, 4(3):243–271, 1973.

[BT76] I. Borosh and L. B. Treybig. Bounds on positive integral solutions of linear diophantine equations. *Proceedings of the American Mathematical Society*, 55(2):299–304, 1976.

[C^+99] P. J. Cameron et al. *Permutation groups*, volume 45. Cambridge University Press, 1999.

[CEFJ96] E. M. Clarke, R. Enders, T. Filkorn, and S. Jha. Exploiting symmetry in temporal logic model checking. *Formal methods in system design*, 9(1-2):77–104, 1996.

[CHVB18] E.M. Clarke, T.A. Henzinger, H. Veith, and R. Bloem, editors. *Handbook of Model Checking*. Springer, 2018.

[Coo72] D. C Cooper. Theorem proving in arithmetic without multiplication. *Machine intelligence*, 7(91-99):300, 1972.

[ES96] E. A. Emerson and A. P. Sistla. Symmetry and model checking. *Formal methods in system design*, 9(1-2):105–131, 1996.

[FPS15] H. Fernau, M. Paramasivan, and M. L. Schmid. Jumping finite automata: characterizations and complexity. In *International Conference on Implementation and Application of Automata*, pages 89–101. Springer, 2015.

[FR74] M.J. Fischer and M.O. Rabin. *Super-exponential Complexity of Presburger Arithmetic*. Project MAC: MAC technical memorandum. Massachusetts Institute of Technology Project MAC, 1974. URL: `https://books.google.co.il/books?id=ijoNHAAACAAJ`.

[Haa18] C. Haase. A survival guide to presburger arithmetic. *ACM SIGLOG News*, 5(3):67–82, 2018. URL: `https://dl.acm.org/citation.cfm?id=3242964`.

[HKR97] T.A. Henzinger, O. Kupferman, and S. Rajamani. Fair simulation. In *Proc. 8th Conferance on Concurrency Theory*, volume 1243 of *Lecture Notes in Computer Science*, Warsaw, July 1997. Springer-Verlag.

[Hof20] S. Hoffmann. State complexity bounds for the commutative closure of group languages. In *International Conference on Descriptional Complexity of Formal Systems*, pages 64–77. Springer, 2020.

[HW87] M. P. Herlihy and J. M. Wing. Axioms for concurrent objects. In *Proceedings of the 14th ACM SIGACT-SIGPLAN symposium on Principles of programming languages*, pages 13–26, 1987.

[ID96] C. N. Ip and D. L. Dill. Better verification through symmetry. *Formal methods in system design*, 9(1-2):41–75, 1996.

[KRS09] J. Kao, N. Rampersad, and J. Shallit. On nfas where all states are final, initial, or both. *Theoretical Computer Science*, 410(47-49):5010–5021, 2009.

[LNRS16] A. W. Lin, T. K. Nguyen, P. Rümmer, and J. Sun. Regular symmetry patterns. In *International Conference on Verification, Model Checking, and Abstract Interpretation*, pages 455–475. Springer, 2016.

[Mil71] R. Milner. An algebraic definition of simulation between programs. In *Proc. 2nd Int. Joint Conf. on Artificial Intelligence*, pages 481–489. British Computer Society, 1971.

[MP19] A. Muscholl and G. Puppis. The Many Facets of String Transducers (Invited Talk). In Rolf Niedermeier and Christophe Paul, editors, *36th International Symposium on Theoretical Aspects of Computer Science (STACS 2019)*, volume 126 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 2:1–2:21, Dagstuhl, Germany, 2019. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. `doi:10.4230/LIPIcs.STACS.2019.2`.

[MZ12] A. Meduna and P. Zemek. Jumping finite automata. *International Journal of Foundations of Computer Science*, 23(07):1555–1578, 2012.

[Opp78] D. C. Oppen. A 222pn upper bound on the complexity of presburger arithmetic. *Journal of Computer and System Sciences*, 16(3):323–332, 1978.

[Par66] R. J. Parikh. On context-free languages. *J. of the ACM*, 13(4):570–581, 1966.

[VSS05] K. N. Verma, H. Seidl, and T. Schwentick. On the complexity of equational horn clauses. In *International Conference on Automated Deduction*, pages 337–352. Springer, 2005.

Appendix A. PSPACE Hardness

Lemma A.1. *Universality of* NFAs *over alphabet $\Sigma = \{0,1\}$, where all states are accepting, and the degree of nondeterminism is at most* 2*, is* PSPACE*-complete.*

Proof. In [KRS09], it is shown that universality of NFAs remains PSPACE-complete even for NFAs over alphabet $\Sigma = \{0,1\}$ and all states accepting. Thus, we only need to show that this remains the case under the restriction that $|\delta(q,\sigma)| \leq 2$ for every state q and letter σ.

To see this, we start by observing that universality remains PSPACE-complete for NFAs over alphabet $\{0,1,\$\}$ with nondeterminism degree at most 2. Indeed, given an NFA over $\{0,1\}$ with maximal nondeterminism degree $d > 2$, we can replace each transition of the form[6] $\delta(q,\sigma) = \{q_1,\ldots,q_d\}$ with a binary tree of depth $\lceil \log d \rceil$, reading \$ on all transitions, which starts at q and ends in $q_1,\ldots,q_d$. Thus, we introduce at most d states for every transition. By marking these states as accepting, this reduction maintains universality, and requires a polynomial blowup.

Next, we observe that the reductions in [KRS09, Lemma 2] first transform an NFA over alphabet size k to an NFA over alphabet size $k+1$ with all states accepting and with identical nondeterminism degree (indeed, the only added transitions are in fact deterministic), and then transforms an NFA with all states accepting and alphabet size 4 to an NFA with all states accepting and alphabet size 2, with an equal nondeterminism degree (essentially by encoding each of the 4 letters as two letters in $\{0,1\}$).

Since we start this chain of reductions with an NFA of nondeterminism degree at most 2, we maintain this property throughout the proof. □

A.1. **Proof of Theorem 4.7.**

We show a reduction from the universality problem for NFAs over alphabet $\{0,1\}$ where all states are accepting and the degree of nondeterminism is at most 2, to round equivalence with $k = 2$ and with Λ given as a DFA of constant size. The former is shown to be PSPACE-hard in Lemma A.1.

Consider an NFA $\mathcal{N} = \langle Q, \{0,1\}, \delta, q_0, Q\rangle$ where $|\delta(q,\sigma)| \leq 2$ for every $q \in Q$ and $\sigma \in \{0,1\}$. We construct two transducers $\mathcal{T}_1$ and $\mathcal{T}_2$ over input and output alphabets $\Sigma_I = \{a,b,c,d\}$ and $\Sigma_O = \{\top,\bot\}$ and $\Lambda \subseteq \Sigma_I^*$, such that $L(\mathcal{N}) = \{0,1\}^*$ iff $\mathcal{T}_1 \equiv_{2,\Lambda} \mathcal{T}_2$.

Set $\Lambda = (ab+cd)^*$ (described as a 4-state DFA). Intuitively, our reduction encodes $\{0,1\}$ into $\{a,b,c,d\}^2$ by setting 0 to correspond to ab and to ba, and 1 to cd and to dc. Then, $\mathcal{T}_1$ keeps outputting $\top$ for all inputs in Λ, thus mimicking "accepting" every word in $\{0,1\}^*$. We then construct $\mathcal{T}_2$ so that every nondeterministic transition of $\mathcal{N}$ on e.g., 0 is replaced by two deterministic branches on ab and on ba. Hence, when we are allowed to permute ab and ba by round equivalence, we capture the nondeterminism of $\mathcal{N}$.

We now proceed to define the reduction formally. We construct $\mathcal{T}_1$ independently of $\mathcal{N}$, as depicted in fig. 14, containing 4 states. For every $x \in \Lambda$ we have $\mathcal{T}_1(x) = \top^{|x|}$, and for every other $x \notin \Lambda$ we have $\mathcal{T}_1(x) = \top^m\bot^{|x|-m}$ where m is the length of the maximal prefix of x in $(ab+cd)^*(a+c+\epsilon)$.

We proceed to construct $\mathcal{T}_2$. We can think of the outgoing transitions from every state q as $\delta(q,0) = \{q^{0,0}, q^{0,1}\}$ and $\delta(q,1) = \{q^{1,0}, q^{1,1}\}$ (unless $\mathcal{N}$ has no outgoing transitions on

[6]We can assume all transitions have degree exactly d by adding redundant transitions

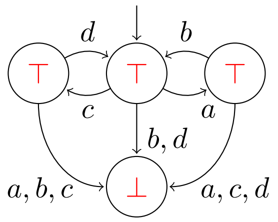


Figure 14: The transducer $\mathcal{T}_1$ in the proof of Theorem 4.7.

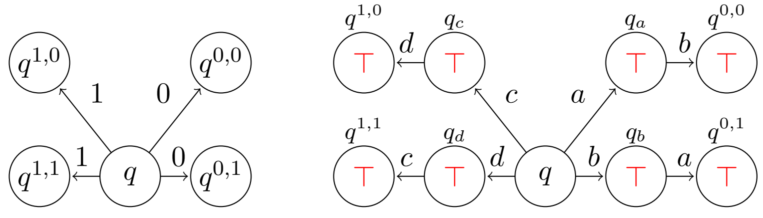


Figure 15: Every state and its 4 transitions in $\mathcal{N}$ (left) turn into 8 transitions in $\mathcal{T}_2$ (right). All transitions not drawn in the right figure lead to $q_\perp$, a sink state labelled $\perp$.

one of the letters, see below). We obtain $\mathcal{T}_2$ from $\mathcal{N}$ by introducing 4 new states q_a, q_b, q_c, q_d for every state $q \in Q$, and setting the transitions and labels as depicted in fig. 15. In case $\mathcal{N}$ does not have a transition on e.g., 0 from q, then instead of going to q_a or q_b, we proceed to a new state $q_\perp$ labelled $\perp$, which is a sink state. In addition, $q_\perp$ is reached upon any transition not yet defined. Observe that for every $x \in \Lambda$ we have $\mathcal{T}_2(x) = \top^m \perp^{|x|-m}$ for some $0 \leq m \leq |x|$ (since $q_\perp$ is a sink).

We now claim that $L(\mathcal{N}) = \{0,1\}^*$ iff $\mathcal{T}_1 \equiv_{2,\Lambda} \mathcal{T}_2$. For the first direction, assume $L(\mathcal{N}) = \{0,1\}^*$. Observe that $\mathcal{T}_2 \prec_{2,\Lambda} \mathcal{T}_1$ independently: for every $x \in (ab+cd)^*$, denote $\mathcal{T}_2(x) = \top^m \perp^{|x|-m}$, then we can construct $x' \asymp_2 x$ such that $\mathcal{T}_1(x') = \top^m \perp^{|x|-m}$ by leaving x unchanged m steps, and then permuting the letters such that the run of $\mathcal{T}_1$ moves to the sink labelled $\perp$ (indeed, observe that m must be even by the construction of $\mathcal{T}_2$, and hence $\mathcal{T}_1$ can permute e.g., ab to ba in order to start outputting $\perp$ on an even step).

Next, we show that $\mathcal{T}_1 \prec_{2,\Lambda} \mathcal{T}_2$. Consider $x \in (ab+cd)^*$, so that $\mathcal{T}_1(x) = \top^{|x|}$, and let $w \in \{0,1\}^*$ be the word obtained from x by identifying ab with 0 and cd with 1. Since $L(\mathcal{N}) = \{0,1\}^*$, there exists a run (and hence an accepting run) of $\mathcal{N}$ on w, denoted $s_0, s_1, \dots, s_n$. We now obtain $x'' \asymp_2 x$ by identifying each letter 0 in x with either ab or ba, and each letter 1 with cd or dc, such that the run of $\mathcal{T}_2$ on x'' simulates the run of $\mathcal{N}$ on w. Thus, $\mathcal{T}_2(x'') = \top^{|x''|}$, and $\mathcal{T}_2(x'') \asymp_2 \mathcal{T}_1(x)$, so we are done.

Conversely, if $\mathcal{T}_1 \equiv_{2,\Lambda} \mathcal{T}_2$, then in particular $\mathcal{T}_1 \prec_{2,\Lambda} \mathcal{T}_2$. We claim that $L(\mathcal{N}) = \{0,1\}^*$. Consider $w \in \{0,1\}^*$. Dually to the above, we obtain from w a word $x \in (ab+cd)^*$ by identifying 0 with ab and 1 with cd, so that $\mathcal{T}_1(x) = \top^{|x|}$. Since $\mathcal{T}_1 \prec_{2,\Lambda} \mathcal{T}_2$, there exists $x' \asymp_2 x$ such that $\mathcal{T}_2(x') = \top^{|x|}$. Observe that x' must be obtained from x by (possibly) changing each ab to ba and each cd to dc. In particular, the run of $\mathcal{T}_2$ on x' induces a run of $\mathcal{N}$ on w by identifying both ab and ba as 0 and both cd and dc as 1. This gives $w \in L(\mathcal{N})$, so $L(\mathcal{N}) = \{0,1\}^*$, which concludes the proof. □

A.2. **Proof of Theorem 5.10.** In order to show that existential round equivalence is PSPACE-hard, we build upon the reduction in the proof of Theorem 4.7: we again show a reduction from the universality problem for NFAs over alphabet $\{0,1\}$ where all states are accepting and the degree of nondeterminism is at most 2 (cf. Lemma A.1).

Consider an NFA $\mathcal{N} = \langle Q, \{0,1\}, \delta, q_0, Q\rangle$ where $|\delta(q,\sigma)| \leq 2$ for every $q \in Q$ and $\sigma \in \{0,1\}$. We construct two transducers $\mathcal{T}_1$ and $\mathcal{T}_2$ over input and output alphabets $\Sigma_I = \{a,b,c,d,\#\}$ and $\Sigma_O = \{\top,\perp\}$ and $\Lambda \subseteq \Sigma_I^*$, such that $L(\mathcal{N}) = \{0,1\}^*$ iff $\mathcal{T}_1 \equiv_{2,\Lambda} \mathcal{T}_2$.

Intuitively, the idea is to use a similar encoding of $\{0,1\}$ in $\{a,b,c,d\}$ whereby 0 corresponds to either ab or ba and 1 to cd or dc. Now, however, since k is not fixed to 2, we also allow arbitrary padding with sequences of ##.

Set $\Lambda = (ab + cd + \#\#)^*$ (given as a 5 state DFA). We construct $\mathcal{T}_1$ and $\mathcal{T}_2$ similarly to the proof of Theorem 4.7, by adding self-cycles of length 2 upon reading ##, from every state except the sink $q_\perp$. See figs. 16 and 17 for an illustration.

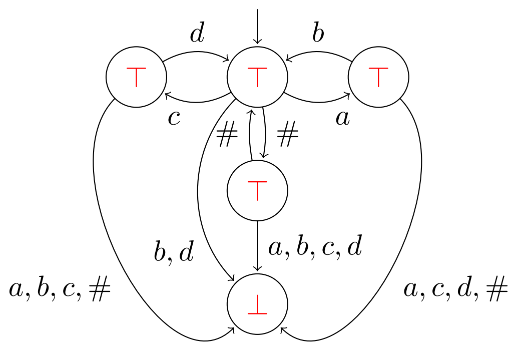


Figure 16: The transducer $\mathcal{T}_1$ in the proof of Theorem 5.10.

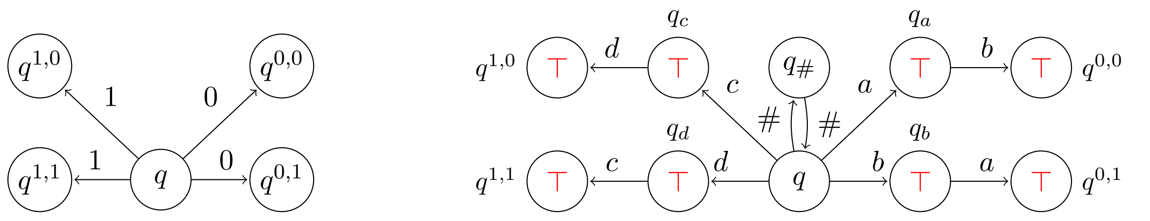


Figure 17: Every state and its 4 transitions in $\mathcal{N}$ (left) turn into 10 transitions in $\mathcal{T}_2$ (right). All transitions not drawn in the right figure lead to $q_\perp$, a sink state labelled $\perp$.

We claim that $L(\mathcal{N}) = \{0,1\}^*$ iff there exists $k > 0$ such that $\mathcal{T}_1 \equiv_{k,\Lambda} \mathcal{T}_2$. For the first direction, assume $L(\mathcal{N}) = \{0,1\}^*$, then we can show that $\mathcal{T}_1 \equiv_{2,\Lambda} \mathcal{T}_2$ by following the proof of Theorem 4.7 line for line, with the addition that blocks of the form ## leave the state of both $\mathcal{T}_1$ and $\mathcal{T}_2$ unchanged.

For the converse direction, assume $\mathcal{T}_1 \equiv_{k,\Lambda} \mathcal{T}_2$, and in fact we only assume $\mathcal{T}_1 \prec_{k,\Lambda} \mathcal{T}_2$ for some $k > 0$. We further assume w.l.o.g. that k is even, otherwise we can just take $2k$ (since we also have $\mathcal{T}_1 \prec_{2k,\Lambda} \mathcal{T}_2$).

Consider $w \in \{0,1\}^*$. We obtain from w a word $x \in (ab + cd + \#\#)^*$ by identifying 0 with $ab\#^{k-2}$ and 1 with $cd\#^{k-2}$. Observe that $\mathcal{T}_1(x) = \top^{|x|}$, and that x is indeed a k-round word in Λ, with each round being either $ab\#^{k-2}$ or $cd\#^{k-2}$.

Since $\mathcal{T}_1 \prec_{k,\Lambda} \mathcal{T}_2$, there exists $x' \asymp_k x$ such that $\mathcal{T}_2(x') = \top^{|x|}$. Observe that x' must be obtained from x by (possibly) changing each ab to ba and each cd to dc, and by shifting the location of this pair within the # symbols. Indeed, otherwise the run of $\mathcal{T}_2$ on x' ends in $q_\perp$. In particular, the run of $\mathcal{T}_2$ on x' induces a run of $\mathcal{N}$ on w by identifying both ab

and ba as 0 and both cd and dc as 1. Thus, $w \in L(\mathcal{N})$, so $L(\mathcal{N}) = \{0,1\}^*$, and the proof is concluded. □

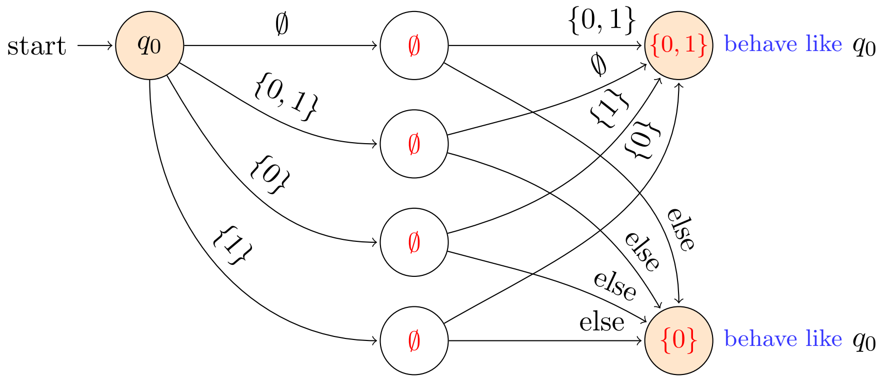
Appendix B. Variants of Round Simulation

We start by presenting some transducers that aid us in the proof of strictness of the remaining notions, all being variants of RR:

(1) RR that expects all requests in the beginning of every round, but outputs like the original (e.g. $\{0,2\}\{1\}\{1\}$ would output $\{0\}\emptyset\{2\}$), modelled by $\mathcal{T}_1$.
(2) RR that expects input as in the original, but outputs all grants in the end of the round (e.g. $\{0,2\}\{1\}\{1\}$ would output $\emptyset\emptyset\{0,1\}$), modelled by $\mathcal{T}_2$.
(3) RR such that every other round begins by considering requests of Process 1 before Process 0 (e.g. $\{0\}\{1\}\emptyset \cdot \{0\}\{1\}\emptyset$ would output $\{0\}\emptyset\emptyset \cdot \emptyset\{1\}\emptyset$), modelled by $\mathcal{T}_3$.

Denote by $\mathcal{T}$ the transducer for RR. It is not difficult to see that $\mathcal{T}_1 \prec^{\mathtt{s},\mathtt{u}} \mathcal{T}$ but $\mathcal{T}_1 \not\prec^{\ell,\mathtt{u}} \mathcal{T}$; that $\mathcal{T}_2 \prec^{\mathtt{u},\mathtt{s}} \mathcal{T}$ but $\mathcal{T}_2 \not\prec^{\mathtt{u},\ell} \mathcal{T}$; and that $\mathcal{T}_3 \prec^{\ell,\ell} \mathcal{T}$ but $\mathcal{T}_3 \not\prec^{\ell,\mathtt{u}} \mathcal{T}$ and $\mathcal{T}_3 \not\prec^{\mathtt{u},\ell} \mathcal{T}$.

Example B.1. The transducers in fig. 18 satisfy $\mathcal{T}_1 \prec_2^{\ell,\mathtt{s}} \mathcal{T}_2$ and $\mathcal{T}_1 \prec_2^{\mathtt{s},\ell} \mathcal{T}_2$. This is proved in table 5, which considers all possible forms of each round and gives round equivalent words $x^\ell \preceq_2^\ell x$ and $x^{\mathtt{s}} \preceq_2^{\mathtt{s}} x$ that satisfy the requirements of the definitions.



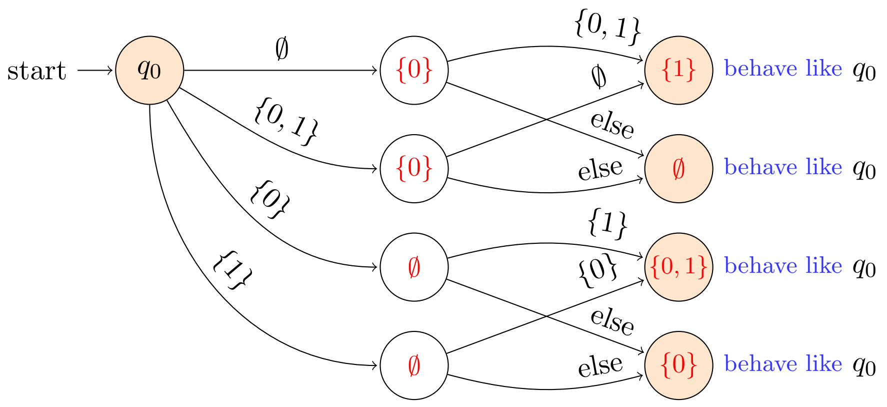

start
q_0
$\emptyset$
$\{0,1\}$
$\{0\}$
$\{1\}$
else
$\{1\}$ behave like q_0
$\emptyset$ behave like q_0
$\{0,1\}$ behave like q_0
$\{0\}$ behave like q_0

Figure 18: Transducers $\mathcal{T}_1$ (up) and $\mathcal{T}_2$ (down) in Example B.1, satisfying $\mathcal{T}_1 \prec_2^{\ell,\mathtt{s}} \mathcal{T}_2$ and $\mathcal{T}_1 \prec_2^{\mathtt{s},\ell} \mathcal{T}_2$, but $\mathcal{T}_1 \not\prec_2^{\ell,\ell} \mathcal{T}_2$. See table 5 for a table summarizing the possible inputs and outputs for $\mathcal{T}_1$.

However, $\mathcal{T}_1 \not\prec^{\ell,\ell}_{k'} \mathcal{T}_2$ for any $k' > 0$. Indeed, consider the word $x = \{0,1\}\emptyset^{k'-1}$ having output $\mathcal{T}_1(x) = \emptyset\{0,1\}\emptyset^{k'-2}$. For $\mathcal{T}_2$ to output the letter $\{0,1\}$, it must see one of the input letters $\{0\}$ and $\{1\}$, since the only state labelled $\{0,1\}$ has two incoming transitions with $\{0\}$ and $\{1\}$. But any $x' \preceq^{\ell}_{k'} x$ will not contain the letters $\{0\}$ and $\{1\}$, so $\mathcal{T}_1(x) \not\preceq^{\ell}_{k'} \mathcal{T}_2(x')$. Therefore $\mathcal{T}_1 \not\prec^{\ell,\ell}_{k'} \mathcal{T}_2$.

Table 5: A table summarizing the outputs of transducer $\mathcal{T}_1$ in Example B.1 on words x of length 2, and round equivalent words x^{ℓ} and x^{s} that satisfy the requirement of x' in the definition of $\mathcal{T}_1 \prec^{\ell,\mathsf{s}}_2 \mathcal{T}_2$ and $\mathcal{T}_1 \prec^{\mathsf{s},\ell}_2 \mathcal{T}_2$.

x	$\mathcal{T}_1(x)$	$x^{\mathsf{s}}\ :\ \mathcal{T}_2(x^{\mathsf{s}}) \preceq^{\mathsf{p}}_2 \mathcal{T}_1(x)$	$\mathcal{T}_2(x^{\mathsf{s}})$	$x^{\mathsf{p}}\ :\ \mathcal{T}_2(x^{\mathsf{p}}) \preceq^{\mathsf{s}}_2 \mathcal{T}_1(x)$	$\mathcal{T}_2(x^{\mathsf{p}})$
$\emptyset\emptyset$	$\emptyset\{0\}$	$\emptyset\emptyset$	$\{0\}\emptyset$	$\emptyset\emptyset$	$\{0\}\emptyset$
$\emptyset\{0\}$	$\emptyset\{0\}$	$\emptyset\{0\}$	$\{0\}\emptyset$	$\emptyset\{0\}$	$\{0\}\emptyset$
$\emptyset\{1\}$	$\emptyset\{0\}$	$\emptyset\{1\}$	$\{0\}\emptyset$	$\emptyset\{1\}$	$\{0\}\emptyset$
$\emptyset\{0,1\}$	$\emptyset\{0,1\}$	$\emptyset\{0,1\}$	$\{0\}\{1\}$	$\{0\}\{1\}$	$\emptyset\{0,1\}$
$\{0\}\emptyset$	$\emptyset\{0\}$	$\{0\}\emptyset$	$\emptyset\{0\}$	$\{0\}\emptyset$	$\emptyset\{0\}$
$\{0\}\{0\}$	$\emptyset\{0\}$	$\{0\}\{0\}$	$\emptyset\{0\}$	$\{0\}\{0\}$	$\emptyset\{0\}$
$\{0\}\{1\}$	$\emptyset\{0,1\}$	$\{0\}\{1\}$	$\emptyset\{0,1\}$	$\{0\}\{1\}$	$\emptyset\{0,1\}$
$\{0\}\{0,1\}$	$\emptyset\{0\}$	$\{0\}\{0,1\}$	$\emptyset\{0\}$	$\{0\}\{0,1\}$	$\emptyset\{0\}$
$\{1\}\emptyset$	$\emptyset\{0\}$	$\{1\}\emptyset$	$\emptyset\{0\}$	$\{1\}\emptyset$	$\emptyset\{0\}$
$\{1\}\{0\}$	$\emptyset\{0,1\}$	$\{1\}\{0\}$	$\emptyset\{0,1\}$	$\{1\}\{0\}$	$\emptyset\{0,1\}$
$\{1\}\{1\}$	$\emptyset\{0\}$	$\{1\}\{1\}$	$\emptyset\{0\}$	$\{1\}\{1\}$	$\emptyset\{0\}$
$\{1\}\{0,1\}$	$\emptyset\{0\}$	$\{1\}\{0,1\}$	$\emptyset\{0\}$	$\{1\}\{0,1\}$	$\emptyset\{0\}$
$\{0,1\}\emptyset$	$\emptyset\{0,1\}$	$\{0,1\}\emptyset$	$\{0\}\{1\}$	$\{0\}\{1\}$	$\emptyset\{0,1\}$
$\{0,1\}\{0\}$	$\emptyset\{0\}$	$\{0,1\}\{0\}$	$\{0\}\emptyset$	$\{0,1\}\{0\}$	$\{0\}\emptyset$
$\{0,1\}\{1\}$	$\emptyset\{0\}$	$\{0,1\}\{1\}$	$\{0\}\emptyset$	$\{0,1\}\{1\}$	$\{0\}\emptyset$
$\{0,1\}\{0,1\}$	$\emptyset\{0\}$	$\{0,1\}\{0,1\}$	$\{0\}\emptyset$	$\{0,1\}\{0,1\}$	$\{0\}\emptyset$