Probabilistic Finite Automaton Emptiness is Undecidable for a Fixed Automaton

Günter Rote

Freie Universität Berlin, Institut für Informatik, Takustraße 9, 14195 Berlin, Germany

Abstract

We construct a probabilistic finite automaton (PFA) with 7 states and an input alphabet of 5 symbols for which the PFA Emptiness Problem is undecidable. The only input for the decision problem is the starting distribution. For the proof, we use reductions from special instances of the Post Correspondence Problem.

We also consider some variations: The input alphabet of the PFA can be restricted to a binary alphabet at the expense of a larger number of states. If we allow a rational output value for each state instead of a yes-no acceptance decision, the number of states can even be reduced to 6.

2012 ACM Subject Classification Theory of computation → Formal languages and automata theory

Keywords and phrases Probabilistic finite automaton, Undecidability, Post Correspondence Problem

Related Version *Conference version at MFCS 2025*: [25]

Contents

1 Probabilistic finite automata (PFA)

A probabilistic finite automaton (PFA) combines characteristics of a finite automaton and a Markov chain. We give a formal definition below. Informally, we can think of a PFA in terms of an algorithm that reads a sequence of input symbols from left to right, having only finite memory. That is, it can manipulate a finite number of variables with bounded range, just like an ordinary finite automaton. In addition, a PFA can make coin flips. As a consequence, the question whether the PFA arrives in an accepting state and thus accepts a given input word is not a yes/no decision, but it happens with a certain probability. The language *recognized* (or *represented*) by a PFA is defined by specifying a probability threshold or *cutpoint* λ. By convention, the language consists of all words for which the probability of acceptance strictly exceeds λ.

The *PFA Emptiness Problem* is the problem of deciding whether this language is empty. This problem is undecidable. Many other undecidability results rely on the undecidability of the PFA Emptiness Problem, for example, problems about matrix products [3], growth problems [4, Chapter 6], or probabilistic planning problems [17].

We present some sharpenings of the undecidability statement, where certain parameters of the PFA are restricted.

1.1 Formal problem definition

Formally, a PFA is given by a sequence of stochastic *transition matrices* M_σ, one for each letter σ from the input alphabet Σ. The states correspond to the rows and columns of the matrices. Thus, the matrices are $d \times d$ matrices for a PFA with d states. The start state is chosen according to a given probability distribution $\pi \in \mathbb{R}^d$. The set of accepting states is characterized by a 0-1-vector $f \in \{0,1\}^d$. In terms of these data, the PFA Emptiness Problem with cutpoint λ is as follows:

> PFA Emptiness. Given a set of k stochastic matrices $\mathcal{M} = \{M_1, \dots, M_k\} \subset \mathbb{Q}^{d\times d}$, a probability distribution $\pi \in \mathbb{Q}^d$, and a 0-1-vector $f \in \{0,1\}^d$, is there a sequence $i_1, i_2, \dots, i_m$ with $1 \le i_j \le k$ for $j = 1, \dots, m$ such that
>
> $$\pi^T M_{i_1} M_{i_2} \dots M_{i_m} f > \lambda \ ? \qquad (1)$$

The most natural choice is $\lambda = \frac{1}{2}$, but the problem is undecidable for any fixed (rational or irrational) cutpoint λ with $0 < \lambda < 1$. We can also ask $\ge \lambda$ instead of $> \lambda$.

1.2 Statement of results

There are three different proofs of the basic undecidability theorem in the literature. The first proof is due to Masakazu Nasu and Namio Honda [20] from 1969.[1] The second proof, by Volker Claus [6], is loosely related to this proof. Both proofs use a reduction from the Post Correspondence Problem (PCP, see Section 1.4). A completely independent proof with a very different approach, namely a reduction from the halting problem for 2-counter machines, was given by Anne Condon and Richard J. Lipton [7] in 1989, based on ideas of Rūsiņš

[1] This proof is commonly misattributed to Paz [23], although Paz gave credit to Nasu and Honda [23, Section IIIB.7, Bibliographical notes, p. 193].

Theorem	π	$\lvert\mathcal{M}\rvert$	$M \in \mathcal{M}$	f	acceptance crit.
Thm. 1a	*input*	4	7×7, positive, *input*	$f = e_1$	$> 1/7$ or $\geq 1/7$
Thm. 1b	*input*	6	7×7, positive, *fixed*	$f = e_1$	$> 1/7$ or $\geq 1/7$
Thm. 1c	*input*	2	18×18, positive, *input*	$f = e_1$	$> 1/18$ or $\geq 1/18$
Thm. 1d	*input*	2	28×28, positive, *fixed*	$f = e_1$	$> 1/28$ or $\geq 1/28$
Thm. 2a	$\pi = e_2$	4	6×6, positive, *input*	$f \in [0,1]^6$, *input*	$> \lambda$ or $\geq \lambda$
Thm. 2b	*fixed*	5	6×6, positive, *fixed*	$f \in [0,1]^6$, *input*	$> \lambda$ or $\geq \lambda$
Claus [6]	$\pi = e_2$	9 (5)	9×9, positive, *input*	$f = e_1$	$> 1/9$
Claus [6]	$\pi = e_2$	2	65×65, positive, *input*	$f = e_1$	$> 1/65$
	$\pi = e_2$	2	(37×37), positive, *input*	$f = e_1$	$> 1/37$
B.&C. [2]	$\pi = e_2$	2	46×46, positive, *input*	$f = e_1$	$> 1/46$ or $\geq 1/46$
	$\pi = e_2$	2	(34×34), positive, *input*	$f = e_1$	$> 1/34$ or $\geq 1/34$
Hirv. [15]	$\pi = e_2$	2	25×25, positive, *input*	$f = e_1$	$> 1/25$
	$\pi = e_2$	2	(20×20), positive, *input*	$f = e_1$	$> 1/20$

Table 1 The main characteristics of the data π, $\mathcal{M}$, and f for different undecidable versions of PFA Emptiness. The vectors e_1 and e_2 are two standard unit vectors of appropriate dimension, indicating a single accepting state or a single deterministic start state. The results under the double line are previous results from the literature, which refer to results of Claus [6, Theorem 6(iii)] from 1981, Blondel and Canterini [2, Theorem 2.1] from 2003, and Hirvensalo [15, Section 3] from 2007. The numbers in parentheses are the figures that would be obtained by basing the proofs on Neary's undecidable 5-word instances of the Post Correspondence Problem (PCP) instead of the smallest known undecidable PCP instances that were current at the time, see [24, Theorems 5 and 6].

Freivalds [10] from 1981. The somewhat intricate history is described in [24, Section 3] as part of an extensive survey of the various undecidability proofs.

PFA Emptiness remains undecidable under various constraints on the number of transition matrices (size of the alphabet) and their size (number of states). The first result in this direction is due to Claus [6] from 1981. Later improvements were made by Blondel and Canterini [2] and Hirvensalo [15], who were apparently unaware of [6] and concentrated on the case of two matrices (*binary* input alphabet). An overview of these results is shown in Table 1 below the double line.

We improve these bounds, concentrating on the minimum number of states without restricting the number of matrices, see Theorem 1. (Such results are only implicit in the proofs of [2] and [15].) Undecidability even holds for a PFA where all data except the starting distribution π are fixed.

We also get improved bounds for the 2-matrix case. For the variation where the output vector f is not restricted to a 0-1-vector, we can reduce the number of states to 6 (Theorem 2).

Our results are stated in the following theorems and summarized in Table 1.

▶ **Theorem 1.** *The PFA Emptiness Problem is undecidable for PFAs with a single accepting state and the following restrictions on the number of transition matrices (size of the input alphabet) and their size (number of states):*

(a) *$\mathcal{M}$ consists of 4 positive doubly-stochastic transition matrices of size 7×7, with cutpoint $\lambda = 1/7$.*

(b) *$\mathcal{M}$ consists of 6 fixed positive doubly-stochastic transition matrices of size 7×7, with cutpoint $\lambda = 1/7$. The only variable input is the starting distribution $\pi \in [0,1]^7$.*

(c) *$\mathcal{M}$ consists of 2 positive transition matrices of size 18×18, with cutpoint $\lambda = 1/18$.*

(d) *$\mathcal{M}$ consists of 2 fixed positive transition matrices of size 28×28, with cutpoint $\lambda = 1/28$.*

All statements hold also for weak inequality ($\geq \lambda$) as the acceptance criterion.

▶ **Theorem 2.** *For any cutpoint λ in the interval $0 < \lambda < 1$, the PFA Emptiness Problem with an output vector f with entries $0 \leq f_i \leq 1$ is undecidable for PFAs with the following restrictions on the number of transition matrices (size of the input alphabet) and their size (number of states):*

(a) *$\mathcal{M}$ consists of 4 positive transition matrices of size 6×6. There is a fixed deterministic start state.*

(b) *$\mathcal{M}$ consists of 5 fixed positive transition matrices of size 6×6. There is a fixed starting distribution π. The only input of the problem is the vector $f \in [0, 1]^6$ of output values.*

All statements hold also for weak inequality ($\geq \lambda$) as the acceptance criterion.

By combining the different proof techniques, one can extend Theorem 2 to PFAs with *two matrices* and a fixed start state or starting distribution, analogous to Theorem 1c–d, but we have not worked out these results.

Some weaker results with fixed matrices were previously obtained in a technical report [24, Theorems 2 and 4]. For example, PFA Emptiness was shown to be undecidable for 52 fixed 9×9 matrices, in the setting of Theorem 2 where the final vector f is the only input, with acceptance criterion $\geq 1/2$. For acceptance with strict inequality ($> 1/4$), 52 fixed matrices of size 11×11 were needed. In these constructions, the PCP was derived from a universal Turing machine. For the case of 2 fixed matrices with a single accepting state, where the start distribution π is the only input, a bound of 572 states was obtained [24, Theorem 3].

Rational-Weighted Automata. A *weighted automaton* over the reals or a *generalized probabilistic automaton* or *pseudo-stochastic automaton* is similar, but the start vector π, the end vector f, and the matrices M_i can be filled with arbitrary real numbers. These automata are in some ways more natural than PFAs, and, in particular with rational weights, they have recently attracted a lot of attention. Since they generalize PFAs, all our undecidability results immediately carry over to rational-weighted automata.

1.3 Contributions of this paper and relations to other problems

The most striking result of the paper is that PFA Emptiness is undecidable already for a *fixed* PFA, where only the starting distribution π, or the output vector f is an input. This follows from a combination of known ideas: The main observation, namely that the reduction of Matiyasevich and Sénizergues [19] from undecidable semi-Thue systems leads to instances of the Post Correspondence problem (PCP) where only the starting pair is variable, has been made by Halava, Harju, and Hirvensalo in 2007 [13, Theorem 6]. The idea of merging the first or last matrix into the starting distribution π or into the final vector f was used by Hirvensalo in 2007 [15, Step 2 in Section 3].

On the technical side, our major contribution is the reduction of the number of states to six, even for PFAs, and even while keeping the matrices fixed, at least for the version where the input is the (fractional) output vector f.

Weighted automata are in some ways more natural than PFAs, and these automata, in particular rational-weighted automata, have recently attracted a lot of attention. Since they generalize PFAs, all our undecidability results immediately carry over to rational-weighted automata.

The PFA Emptiness Problem is a problem about matrix products. There is a great variety of problems about matrix products whose undecidability has been studied; see [8] for a recent survey of this rich area. In fact, the first problem outside the fields of mathematical logic and theory of computing that was shown to be undecidable is a problem about matrix products: Markov proved in 1947 that it is undecidable whether the semigroups generated two finite sets of matrices contain a common element [18]; see Halava and Harju [12] for a presentation of Markov's proof (as well as an alternative proof). Our basic approach is the same as Markov's: to model the Post Correspondence Problem (PCP) by matrix products. Our particular technique for doing this has its roots in Paterson's undecidability proof [22] from 1970 of the *matrix mortality problem* for 3×3 matrices.

Besides, we have brought to the light some papers that were apparently forgotten, like Nasu and Honda's original undecidability proof from 1969 [20], and Claus [6]. Also, our technique for converting matrices with row sums 0 to positive matrices with row sums 1 (hence stochastic matrices) in Section 2.6 is more streamlined and elegant than the proofs that we have seen in the literature.

1.4 The Post Correspondence Problem (PCP)

In the *Post Correspondence Problem* (PCP), we are given a list of pairs of words (v_1, w_1), $(v_2, w_2), \dots, (v_k, w_k)$. The problem is to decide if there is a nonempty sequence $i_1 i_2 \dots i_m$ of indices $i_j \in \{1, 2, \dots, k\}$ such that

$$v_{i_1} v_{i_2} \dots v_{i_m} = w_{i_1} w_{i_2} \dots w_{i_m}$$

This is one of the well-known undecidable problems. According to Neary [21], the PCP is already undecidable with as few as five word pairs.

2 Proofs of Theorem 1a and Theorem 1b (few states)

We follow the same overall proof strategy as Claus [6], Blondel and Canterini [2] and Hirvensalo [15]: They use undecidable PCP instances with few word pairs and transform them to the Emptiness Problem for *integer-weighted* automata, which are then converted to PFAs. We deviate from this approach by using an automaton with *fractional* weights (Section 2.1). These matrices can be converted to column-stochastic matrices without the overhead of an extra state (Section 2.4).

The proof of Theorem 1a contains the main ideas. For the reduction to two matrices, we then apply a technique of Hirvensalo [15] (Section 3.1). All other results are obtained by slight variations of these methods in combination with appropriate results from the literature.

2.1 Step 1: Modeling word pairs by matrices

For a string $u = u_1 u_2 \dots u_n$ of decimal digits, we denote its fractional decimal value by $0.u = \sum_{j=1}^{n} u_j \cdot 10^{-j}$. For example, if $u = \texttt{432100}$, then $0.u = 0.4321$. We will take care to avoid trailing zeros, because their disappearance could cause problems.

For two strings v, w of digits in $\{\texttt{11}, \texttt{12}\}^*$, we define the matrix

$$A_0(v,w) = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0.v & 10^{-|v|} & 0 & 0 & 0 & 0 \\ (0.v)^2 & 2 \cdot 10^{-|v|} \cdot 0.v & 10^{-2|v|} & 0 & 0 & 0 \\ 0.w & 0 & 0 & 10^{-|w|} & 0 & 0 \\ (0.w)^2 & 0 & 0 & 2 \cdot 10^{-|w|} \cdot 0.w & 10^{-2|w|} & 0 \\ 0.v \cdot 0.w & 10^{-|v|} \cdot 0.w & 0 & 10^{-|w|} \cdot 0.v & 0 & 10^{-|v|-|w|} \end{pmatrix}.$$

It is not straightforward to see, but it can be checked by a simple calculation that the matrices $A_0(v,w)$ satisfy a multiplicative law (see Appendix C for a computer check):

▶ **Lemma 3** (Multiplicative Law).

$$A_0(v_1,w_1)A_0(v_2,w_2) = A_0(v_1v_2, w_1w_2) \tag{2}$$

Now we transform these matrices $A_0(v,w)$ by a similarity transformation with the transformation matrix

$$U = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ \frac{1}{99} & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{99}{105} & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}, \quad U^{-1} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ -\frac{1}{99} & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{105}{99} & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}. \tag{3}$$

The resulting matrices $A(v,w) = U^{-1}A_0(v,w)U$ differ in some entries of the first column and the fifth row:

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0.v & 10^{-|v|} & 0 & 0 & 0 & 0 \\ (0.v)^2 + \frac{1}{99}(1{-}10^{-2|v|}) & 2\cdot 10^{-|v|}\cdot 0.v & 10^{-2|v|} & 0 & 0 & 0 \\ 0.w & 0 & 0 & 10^{-|w|} & 0 & 0 \\ \frac{99}{105}\cdot(0.w)^2 & 0 & 0 & \frac{198}{105}\cdot 10^{-|w|}\cdot 0.w & 10^{-2|w|} & 0 \\ 0.v\cdot 0.w & 10^{-|v|}\cdot 0.w & 0 & 10^{-|w|}\cdot 0.v & 0 & 10^{-|v|-|w|} \end{pmatrix}$$

Since the matrices were obtained by a similarity, they satisfy the same multiplicative law:

$$A(v_1,w_1)A(v_2,w_2) = A(v_1v_2, w_1w_2) \tag{4}$$

With the vectors $\pi_1 = (\frac{1}{99}, 0, -1, 0, -\frac{105}{99}, 2)$ and $f_1 = (1,0,0,0,0,0)^T$, we obtain

$$\begin{aligned} \pi_1 A(v,w) f_1 &= \tfrac{1}{99} - (0.v)^2 - \tfrac{1}{99} + 10^{-2|v|}/99 - (0.w)^2 + 2\cdot 0.v\cdot 0.w \\ &= -(0.v - 0.w)^2 + 10^{-2|v|}/99 \end{aligned} \tag{5}$$

The sign of this expression can detect inequality of v and w, as we will presently show.

We could have taken the simpler matrix A_0 with $\pi_0 = (0,0,-1,0,-1,2)$, and this would give $\pi_0 A_0(v,w) f_1 = -(0.v - 0.w)^2$. The contortions with the matrix U were necessary to generate a tiny positive term in (5). The reason for the peculiar entries $\frac{99}{105}$ and $\frac{105}{99}$ in (3) and in π_1 will become apparent after the next transformation.

2.2 Equality detection

▶ **Lemma 4** (Equality Detection Lemma). *Let $v, w \in \{11, 12\}^*$.*

1. *If $v = w$, then $(0.v - 0.w)^2 - 10^{-2|v|}/99 < 0$.*

2. *If $v \neq w$, then $(0.v - 0.w)^2 - 10^{-2|v|}/99 > 0$.*

In particular, $(0.v - 0.w)^2 - 10^{-2|v|}/99$ is never zero.

Proof. The first statement is obvious.

To see the second statement, we first consider the case that one of the strings is a prefix of the other (Case A): If w is a prefix of v, then for fixed w, the smallest possible difference $|0.v - 0.w|$ among the strings v that extend w is achieved when $0.v = 0.w11$.

Similarly, if v is a prefix of w, then the smallest possible difference is achieved when $0.w = 0.v\texttt{11}$. In either case, $|0.v - 0.w| \geq 10^{-\min\{|v|,|w|\}} \cdot 0.11 \geq 10^{-|v|} \cdot 0.11$. Thus, $(0.v - 0.w)^2 \geq 10^{-2|v|} \cdot 0.0121 > 10^{-2|v|} \cdot 0.010101\ldots = 10^{-2|v|}/99$.

Case B: Neither of v and w is a prefix of the other. Suppose v and w share k leading digit pairs $u \in \{\texttt{11}, \texttt{12}\}^k$, $0 \leq k < |v|/2$. Then one of the two strings starts with $u\texttt{12}$ and the other with $u\texttt{11}$; the smallest difference $|0.v - 0.w|$ between two such numbers is achieved between $0.u\texttt{12}$ and $0.u\texttt{11121212}\ldots$, and thus $|0.v - 0.w| > 10^{-2k} \cdot 0.00878787\ldots > 10^{-2k} \cdot 0.005 \geq 10^{-|v|+2} \cdot 0.005 = 10^{-|v|}/2$. After squaring this relation, the claim follows with an ample margin. ◀

2.3 Modeling the Post Correspondence Problem (PCP)

The multiplicative law (4) and the capacity to detect string equality are all that is needed to model the PCP.

In the PCP, it is no loss of generality to assume that the words in the pairs (v_i, w_i) use a binary alphabet, since any alphabet can be coded in binary. We recode them to the binary "alphabet" $\{\texttt{11}, \texttt{12}\}$, i.e., they become words in $\{\texttt{11}, \texttt{12}\}^*$ of doubled length. We form the matrices $A_i = A(v_i, w_i)$. Multiplicativity gives the relation $A_{i_1} A_{i_2} \ldots A_{i_m} = A(v_{i_1} v_{i_2} \ldots v_{i_m}, w_{i_1} w_{i_2} \ldots w_{i_m})$, and by Lemma 4, $\pi_1 A_{i_1} A_{i_2} \ldots A_{i_m} f_1 > 0$ if and only if $v_{i_1} v_{i_2} \ldots v_{i_m} = w_{i_1} w_{i_2} \ldots w_{i_m}$, i.e., $i_1 i_2 \ldots i_m$ is a solution of the PCP.

Since the value 0 is excluded by Lemma 4, nothing changes if the condition "> 0" is replaced by "≥ 0." This property will propagate through the proof, with the consequence that in the resulting theorems, it does not matter if we take $> \lambda$ or $\geq \lambda$ as the acceptance criterion for the PFA. We will not mention this issue any more and work only with strict inequality.

There are two problems that we still have to solve:

- The empty sequence ($m = 0$) always fulfills the inequality although it does not count as a solution of the PCP.
- The matrices A_i are not stochastic, and π_1 is not a probability distribution. So far, what we have is a *generalized probabilistic automaton*, or *rational-weighted automaton*.

Turakainen [27] showed in 1969 that a generalized probabilistic automaton can be converted to a PFA without changing the recognized language (with the possible exception of the empty word), and this can even be done by adding only two more states [28, Theorem 1(i)]. Thus, by the general technique of [28], we can get a PFA with 8 states. We will use some tailored version of this method, involving nontrivial techniques, so that no extra state is needed to make the matrices stochastic and to simultaneously get rid of the empty solution.

History of ideas. The idea of modeling of the PCP by multiplication of integer matrices was pioneered by Markov [18] in 1947. He used the matrices $\left(\begin{smallmatrix}1 & 0\\ 1 & 1\end{smallmatrix}\right)$ and $\left(\begin{smallmatrix}1 & 1\\ 0 & 1\end{smallmatrix}\right)$, which generate a free semigroup. A different representation, closer to the one we are using, goes back to Paterson [22] in 1970, who used it to show that mortality for 3×3 matrices is undecidable.

Our matrix $A_0(v, w)$ is a variation of the integer matrices that were proposed by Volker Claus [6] in 1981, and again by Blondel and Canterini [2, p. 235] in 2003, in the very context of constructing small undecidable instances of the PFA Emptiness Problem. These matrices extend Paterson's matrices to larger matrices that can produce quadratic terms. They use *positive* powers of the base 10 and *integer* decimal values $(u)_{10}$ of the strings u instead of fractional values $0.u$. Such matrices satisfy a multiplicative law such as (2) but with a reversal of the factors. In fact, the method works equally well for any other radix instead of 10.[2]

[2] "The notation is quaternary, decimal, etc., according to taste." (Paterson [22]). Claus [6] used radix 3.

Our novel idea is to use this construction with *negative* powers of 10, leading to entries that are roughly at the same scale as a stochastic matrix, thus facilitating further transformations.

2.4 Step 2: Making column sums 1

We apply another similarity transformation, using the matrix

$$V = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}, \ V^{-1} = \begin{pmatrix} 1 & -1 & -1 & -1 & -1 & -1 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix},$$

but this time, we also transform the vectors π_1 and f_1, leaving the overall result unchanged:

$$\begin{aligned} \pi_1 A_{i_1} A_{i_2} \dots A_{i_m} f_1 &= (\pi_1 V)(V^{-1} A_{i_1} V)(V^{-1} A_{i_2} V) \dots (V^{-1} A_{i_m} V)(V^{-1} f_1) \\ &= \pi_2 B_{i_1} B_{i_2} \dots B_{i_m} f_2 \end{aligned} \tag{6}$$

We analyze the effect of the similarity transform for a matrix of the general form

$$A = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ a_{21} & a_{22} & 0 & 0 & 0 & 0 \\ a_{31} & a_{32} & a_{33} & 0 & 0 & 0 \\ a_{41} & a_{42} & a_{43} & a_{44} & 0 & 0 \\ a_{51} & a_{52} & a_{53} & a_{54} & a_{55} & 0 \\ a_{61} & a_{62} & a_{63} & a_{64} & a_{65} & a_{66} \end{pmatrix}.$$

The transformed matrix is

$$B = V^{-1} A V =$$

$$\begin{pmatrix} K & K_{12} & K - a_{33} - a_{43} - a_{53} - a_{63} & K - a_{44} - a_{54} - a_{64} & K - a_{55} - a_{65} & K - a_{66} \\ a_{21} & a_{21} + a_{22} & a_{21} & a_{21} & a_{21} & a_{21} \\ a_{31} & a_{31} + a_{32} & a_{31} + a_{33} & a_{31} & a_{31} & a_{31} \\ a_{41} & a_{41} + a_{42} & a_{41} + a_{43} & a_{41} + a_{44} & a_{41} & a_{41} \\ a_{51} & a_{51} + a_{52} & a_{51} + a_{53} & a_{51} + a_{54} & a_{51} + a_{55} & a_{51} \\ a_{61} & a_{61} + a_{62} & a_{61} + a_{63} & a_{61} + a_{64} & a_{61} + a_{65} & a_{61} + a_{66} \end{pmatrix}$$

with $K = 1 - a_{21} - a_{31} - a_{41} - a_{51} - a_{61}$ and $K_{12} = K - a_{22} - a_{32} - a_{42} - a_{52} - a_{62}$.

▶ **Lemma 5.** **1.** *the matrix $B = V^{-1} A V$ has column sums* 1.
2. *For $v \neq \epsilon$ and $w \neq \epsilon$, the matrix $V^{-1} A(v, w) V$ is positive, and therefore column-stochastic.*

Proof. The first statement can be easily checked directly, but it has a systematic reason:

$$(1,1,1,1,1,1) V^{-1} A V = (1,0,0,0,0,0) A V = (1,0,0,0,0,0) V = (1,1,1,1,1,1)$$

The second statement is not needed for Theorem 1, because positivity is established anyway in Step 4, after destroying it in Step 3, but we need it for Theorem 2. So let us check it: The only entries that are in danger of becoming negative are the entries of the first row. The two entries $a_{21} = 0.v$ and $a_{41} = 0.w$ of the matrix $A(v, w)$ can be as large as $0.121212\ldots \leq 0.15$; all other entries are safely below 0.05. Thus, even the “most dangerous” candidate K_{12}, where 10 of the entries a_{ij} are subtracted from 1, cannot be zero or negative.

It can be checked that rows 2–6 are positive, because the first column of $A(v, w)$, which is positive, has been added to every other column.

Note that the assumptions $v \neq \epsilon$ and $w \neq \epsilon$ are essential; without these assumptions, we would get negative entries. If, for example, $v = \epsilon$ and $w \neq \epsilon$, then $a_{22} = 10^{-|v|} = 1$ and $a_{41} = 0.w > 0$, so $K_{12} = 1 - a_{41} - a_{22} - \cdots$ would be negative. ◀

In the transformation from π_1 to $\pi_2 = \pi_1 V$, the first entry is added to all other entries. Thus, the vector $\pi_1 = (\frac{1}{99}, 0, -1, 0, -1 - \frac{6}{99}, 2)$, becomes $\pi_2 = (\frac{1}{99}, \frac{1}{99}, -1 + \frac{1}{99}, \frac{1}{99}, -1 - \frac{5}{99}, 2 + \frac{1}{99})$, whose entries sum to zero:

$$\pi_2 \begin{pmatrix}1\\1\\1\\1\\1\\1\end{pmatrix} = \pi_1 V \begin{pmatrix}1\\1\\1\\1\\1\\1\end{pmatrix} = \pi_1 \begin{pmatrix}6\\1\\1\\1\\1\\1\end{pmatrix} = 0 \tag{7}$$

It was for this reason that the entry $\frac{105}{99}$ of π_1 and the corresponding entries of the transformations (3) were chosen.

The output vector f is unchanged by the transformation: $f_2 = V^{-1} f_1 = f_1$.

2.5 Step 3: Making row sums 1 with an extra state

By adding an extra column r_i, we now make all row sums equal to 1. We also add an extra row, resulting in a 7×7 matrix

$$C_i = \begin{pmatrix} B_i & r_i \\ 0 & 1 \end{pmatrix} \tag{8}$$

The entries of r_i may be negative. They lie in the range $-5 \le r \le 1$. All column sums are still 1: Since the row sums are 1, the total sum of the entries is 7. Therefore, since the first six column sums are 1, the sum of the last column must also be 1.

The vectors π_2 and f_2 are extended to vectors π_3 and f_3 of length 7 by adding a 0, but they are otherwise unchanged. Thus, the extra column and row of C_i plays no role for the value of the product (6), which remains unchanged:

$$\pi_3 C_{i_1} C_{i_2} \ldots C_{i_m} f_3 = \pi_2 B_{i_1} B_{i_2} \ldots B_{i_m} f_2$$

2.6 Step 4: Making the matrices positive, and hence stochastic

Let J be the doubly-stochastic 7×7 transition matrix of the "completely random transition" with all entries 1/7. Then, with $\alpha = 0.01$, we form the matrices $D_i := (1 - \alpha)J + \alpha C_i$. The constant α is small enough to ensure that $D_i > 0$. Hence the matrices D_i are doubly-stochastic.

If expand of the product $\pi_3 \prod_{j=1}^{m} D_{i_j} = \pi_3 \prod_{j=1}^{m} \big((1 - \alpha)J + \alpha C_{i_j}\big)$, we get a sum of 2^m terms, each containing a product of m of the matrices J or C_i. We find that all terms containing the factor J vanish. The reason is that, since C_i has row and column sums 1, $JC_i = C_i J = J$. Moreover, $\pi_3 J = 0$ by (7). It follows that

$$\pi_3 D_{i_1} D_{i_2} \ldots D_{i_m} f_3 = \alpha^m \cdot \pi_3 C_{i_1} C_{i_2} \ldots C_{i_m} f_3$$

The factor α^m plays no role for the sign, and hence,

$$\pi_3 D_{i_1} D_{i_2} \ldots D_{i_m} f_3 > 0 \tag{9}$$

if and only if $i_1 i_2 \ldots i_m$ is a solution of the PCP.

2.7 Step 5: Turning π into a probability distribution

The start vector π_3 has sum 0 and contains negative entries, which are smaller than -1 but not smaller than -2. We form $\pi_4 = \big((2,2,2,2,2,2,2)+\pi_3)\big)/14$. It is positive and has sum 1. The effect of substituting π_3 by π_4 in (9) is that the result is increased by 1/7. The reason is that $(1,1,1,1,1,1,1)D_{i_1}D_{i_2}\dots D_{i_m} = (1,1,1,1,1,1,1)$, since the matrices D_i are doubly-stochastic, and in particular, column-stochastic, and therefore $(2,2,2,2,2,2,2)/14 \cdot D_{i_1}D_{i_2}\dots D_{i_m}f_3 = 2/14 = 1/7$.

In summary, we have now constructed a true PFA with seven states that models the PCP:

$$\pi_4 D_{i_1}D_{i_2}\dots D_{i_m}f_3 > 1/7 \tag{10}$$

if and only if $i_1 i_2 \dots i_m$ is a (possibly empty) solution of the PCP.

2.8 Using a small PCP

We base our proof on the undecidability of the PCP with 5 word pairs, as established by Turlough Neary [21, Theorem 11] in 2015. Neary constructed PCP instances with five pairs (v_1, w_1), (v_2, w_2), (v_3, w_3), (v_4, w_4), (v_5, w_5) that have the following property: Every solution necessarily starts with the pair (v_1, w_1) and ends with (v_5, w_5). We can also assume that the end pair (v_5, w_5) is used nowhere else. (More precisely, in every *primitive* solution (which is not a concatenation of smaller solutions), the end pair (v_5, w_5) *cannot* appear anywhere else than in the final position.) However, the start pair (v_1, w_1) is also used in the middle of the solutions. (This multipurpose usage of the start pair is one of the devices to achieve such a remarkably small number of word pairs.)[3]

Reversing the words. For our construction it is preferable to have the word pair (v_5, w_5) at the beginning. We thus reverse all words. Any solution sequence $i_1 i_2 \dots i_m$ for the original problem must be also be reversed to $i_m i_{m-1} \dots i_1$, but this does not affect the solvability of the PCP. Thus, we work with PCP instances of the form $(v_1^R, w_1^R), (v_2^R, w_2^R), \dots, (v_5^R, w_5^R)$ with the following property: Every solution sequence $i_1 i_2 \dots i_m$ must start with the pair (v_5^R, w_5^R), and the pair (v_5^R, w_5^R) cannot be used anywhere else: $i_1 = 5$ and $1 \le i_j \le 4$ for $j = 2, \dots, m$.

2.9 Step 6. Merging the leftmost matrix into the starting distribution

We apply the above construction steps to the word pairs $(v_1^R, w_1^R), \dots, (v_5^R, w_5^R)$, leading to five matrices D_1, D_2, D_3, D_4, D_5. Since the leftmost matrix must be D_5 in any solution, we can combine this matrix D_5 with the starting distribution π_4:

$$\pi_4 D_5 D_{i_2}\dots D_{i_m} f_3 = \pi_5 D_{i_2}\dots D_{i_m} f_3,$$

[3] The proof of Theorem 11 in Neary [21] contains an error, but this error can be fixed: His PCP instances encode *binary tag systems.* When showing that the PCP solution must follow the intended patters of the simulation of the binary tag system, Neary [21, p. 660] needs to show that the end pair $(v_5, w_5) = (10^\beta 1111, 1111)$ cannot be used except to bring the two strings to a common end. He claims that a block 1111 cannot appear in the encoded string because in u (the unencoded string of the binary tag system, which is described in Lemma 9) we cannot have two c symbols next to each other. This is not true. The paper contains plenty of examples, and they contradict this claim; for example, the string u' in (7) [21, p. 657] contains seven c's in a row. The mistake can be fixed by taking a longer block of 1s: Looking at the appendants in Lemma 9, it is clear that every block of length $|u|+1$ must contain a symbol b. Thus, the pair $(v_5, w_5) = (10^\beta 1^{|u|+99}, 1^{|u|+99})$ will work as end pair.

leading to a new starting distribution $\pi_5 := \pi_4 D_5$. The matrix D_5 can be removed from the pool $\mathcal{M}$ of matrices, leaving only 4 matrices. We have simultaneously eliminated the empty solution with a product of $m = 0$ matrices. This concludes the proof of Theorem 1a. ◀

2.10 Proof of Theorem 1b (fixed transition matrices) by using a PCP with only one variable word pair

Matiyasevich and Sénizergues [19] constructed PCP instances with seven word pairs (v_1, w_1), (v_2, w_2), (v_3, w_3), (v_4, w_4), (v_5, w_5), (v_6, w_6), (v_7, w_7) that have the following property: Every solution necessarily starts with the pair (v_1, w_1) and ends with (v_2, w_2). Both the start pair (v_1, w_1) and the end pair (v_2, w_2) can be assumed to appear nowhere else, in the sense described in Section 2.8, i. e., apart from the possibility to concatenate solutions to obtain longer solutions. Matiyasevich and Sénizergues [19] used a reduction, due to Claus [5], from the *individual accessibility problem* (or individual word problem) for semi-Thue systems. They showed that the individual accessibility problem is already undecidable for a *particular* semi-Thue system with 3 rules [19, Theorem 3]. Halava, Harju, and Hirvensalo [13, Theorem 6] observed an important consequence of this: One can fix all words except v_1, and leave only v_1 as an input to the problem, and the PCP is still undecidable.

From these word pairs, we form the matrices $A_i = A(v_i, w_i)$ from the words without reversal. Following the same steps as above, we eventually arrive at corresponding matrices $D_1, \ldots, D_7$. We merge D_1 with π_4 into a new starting distribution $\pi_5 := \pi_4 D_1$. We are left with a pool $\mathcal{M} = \{D_2, \ldots, D_7\}$ of six fixed matrices. The only variable input is the starting distribution π_5. This concludes the proof of Theorem 1b. ◀

3 Proofs of Theorem 1c and Theorem 1d (binary input)

3.1 Reduction to two matrices

The following lemma and its proof is extracted from Step 3 in Hirvensalo [15, Section 3]. We denote by $\phi(u)$ the acceptance probability of a (generalized) PFA for an input u, i.e., the value of the product in the expression (1).

▶ **Lemma 6.** *Let $A = (\pi, \mathcal{M}, f)$ be a generalized PFA with $k = |\Sigma| \geq 3$ matrices M_i of size $d \times d$, such that the first row of each matrix M_i is $(1, 0, \ldots, 0)$.*

Then one can obtain a generalized PFA $A' = (\pi', \mathcal{M}', f')$ over the two-symbol alphabet $\{\mathtt{a}, \mathtt{b}\}$, with 2 matrices $M'_\mathtt{a}$ and $M'_\mathtt{b}$ of dimension $(k-1)(d-1)+1$ such that for every word $u' \in \{\mathtt{a}, \mathtt{b}\}^*$ there exists a word $u \in \Sigma^*$ with

$$\phi(u) = \phi'(u'), \tag{11}$$

and conversely, for every word $u \in \Sigma^*$ there exists a word $u' \in \{\mathtt{a}, \mathtt{b}\}^*$ with (11).

If the given matrices M_i are nonnegative, then so are $M'_\mathtt{a}$ and $M'_\mathtt{b}$. If the given matrices M_i are stochastic, then so are $M'_\mathtt{a}$ and $M'_\mathtt{b}$. If π is a probability distribution, then so is π'. The entries of f' are taken from the entries of f.

Thus, if A is a PFA, then B is a PFA as well.

The lemma is not specific about the word u or u' whose existence is guaranteed. Nevertheless, regardless of how these words are found, the statement is sufficient in the context of the emptiness question.

However, we can be explicit about u and u': The construction uses a binary encoding $\tau\colon \Sigma^* \to \{\mathtt{a},\mathtt{b}\}^*$ with the prefix-free codewords $\{\mathtt{b}, \mathtt{ab}, \mathtt{aab}, \dots, \mathtt{a}^{k-2}\mathtt{b}, \mathtt{a}^{k-1}\}$. Then, we can take $u' = \tau(u)$, because

$$\phi(u) = \phi'(\tau(u)).$$

For words u' that are not of the form $\tau(u)$, (11) holds for the longest word u such that $\tau(u)$ is a prefix of u'. In other words, u is the decoding of the longest decodable prefix of u'.

Proof. The procedure is easiest to describe if we assume that A is a PFA. Then we can think of A' as an automaton that decodes the input $u' \in \{\mathtt{a},\mathtt{b}\}^*$ and carries out a transition of A whenever it has read a complete codeword. In addition to the state $q \in \{1,\dots,d\}$ of A, A' needs to maintain a counter i in the range $0 \le i \le k-2$ for the number of $\mathtt{a}$'s of the current partial codeword. Whenever A' reads a $\mathtt{b}$, or if it has read the $(k-1)$-st $\mathtt{a}$, A' performs the appropriate random transition and resets the counter. The number of states of A' is $d \times (k-1)$.

The fact that state 1 is an absorbing state allows a shortcut: If we are in state 1, we can stop to maintain the counter i. Thus, only states $2,\dots,d$ need to be multiplied by $k-1$, and the resulting number of states reduces to $(d-1)(k-1)+1$.

We now describe the construction of A' in terms of transition matrices. This construction is valid also when A is a generalized PFA, where the above description in terms of random transitions makes no sense. To make the description more concrete, we illustrate it with $k = 5$ matrices $M_1,\dots,M_5$ and the corresponding binary codewords $\mathtt{b}$, $\mathtt{ab}$, $\mathtt{aab}$, $\mathtt{aaab}$, $\mathtt{aaaa}$.

We split the $d \times d$ matrices M_i and the vectors π and f into blocks of size $1 + (d-1)$:

$$M_i = \begin{pmatrix} 1 & 0 \\ c_i & \hat{C}_i \end{pmatrix}, \ \pi = \begin{pmatrix} p_1 \\ \hat{\pi} \end{pmatrix}^T, \ f = \begin{pmatrix} f_1 \\ \hat{f} \end{pmatrix}.$$

We define new transition matrices and start and end vectors in block form with block sizes $1 + (d-1) + (d-1) + (d-1) + (d-1)$ as follows:

$$M'_{\mathtt{a}} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & I & 0 & 0 \\ 0 & 0 & 0 & I & 0 \\ 0 & 0 & 0 & 0 & I \\ c_5 & \hat{C}_5 & 0 & 0 & 0 \end{pmatrix}, \ M'_{\mathtt{b}} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ c_1 & \hat{C}_1 & 0 & 0 & 0 \\ c_2 & \hat{C}_2 & 0 & 0 & 0 \\ c_3 & \hat{C}_3 & 0 & 0 & 0 \\ c_4 & \hat{C}_4 & 0 & 0 & 0 \end{pmatrix}, \ \pi' = \begin{pmatrix} p_1 \\ \hat{\pi}_2 \\ 0 \\ 0 \\ 0 \end{pmatrix}^T, \text{ and } f' = \begin{pmatrix} f_1 \\ \hat{f} \\ \hat{f} \\ \hat{f} \\ \hat{f} \end{pmatrix}.$$

From the sequence of powers of $M'_{\mathtt{a}}$,

$$(M'_{\mathtt{a}})^2 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & I & 0 \\ 0 & 0 & 0 & 0 & I \\ c_5 & \hat{C}_5 & 0 & 0 & 0 \\ c_5 & 0 & \hat{C}_5 & 0 & 0 \end{pmatrix}, \ (M'_{\mathtt{a}})^3 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & I \\ c_5 & \hat{C}_5 & 0 & 0 & 0 \\ c_5 & 0 & \hat{C}_5 & 0 & 0 \\ c_5 & 0 & 0 & \hat{C}_5 & 0 \end{pmatrix},$$

we can recognize the pattern of development. We can then work out the matrices $M'_{\mathtt{b}}$, $M'_{\mathtt{a}}M'_{\mathtt{b}}$, $M'_{\mathtt{a}}M'_{\mathtt{a}}M'_{\mathtt{b}}$, $M'_{\mathtt{a}}M'_{\mathtt{a}}M'_{\mathtt{a}}M'_{\mathtt{b}}$, and $M'_{\mathtt{a}}M'_{\mathtt{a}}M'_{\mathtt{a}}M'_{\mathtt{a}}$ and check that they are of the form

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ c_i & \hat{C}_i & 0 & 0 & 0 \\ * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \end{pmatrix}$$

for $i = 1, 2, 3, 4, 5$, having the original matrices M_i in their upper-left corner and otherwise zeros in the first two rows of blocks. Thus, they simulate the original automaton on the states $1, \dots, d$: It is easy to establish by induction that multiplying the initial distribution vector π' with a sequence of such matrices will produce a vector x' of the form

$$x' = (x_1 \ \hat{x} \ 0 \ 0 \ 0) \tag{12}$$

whose first d entries $x = (x_1 \ \hat{x})$ coincide with the entries of the corresponding vector produced with the original start vector π and the original matrices M_i. If x' is multiplied with f', the result $x_1 f_1 + \hat{x}^T \hat{f}$ is the same as with x and the original vector f.

One technicality remains to be discussed: Some "unfinished" words in $\{\mathtt{a}, \mathtt{b}\}^*$ do not factor into codewords but end in a partial codeword a, aa, or aaa. To analyze the corresponding matrix products, we look at the powers $(M'_{\mathtt{a}})^i$, for $i = 1, 2, 3$. They have the form

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & I & 0 & 0 \\ * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \end{pmatrix}, \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & I & 0 \\ * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \end{pmatrix}, \text{ and } \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & I \\ * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \end{pmatrix}. \tag{13}$$

and therefore, multiplying the vector x from (12) with them yields $(x_1 \ 0 \ \hat{x} \ 0 \ 0)$, $(x_1 \ 0 \ 0 \ \hat{x} \ 0)$, and $(x_1 \ 0 \ 0 \ 0 \ \hat{x})$, respectively. If this is multiplied with f', the result is the same as with the vector x' in (12). Thus, as claimed after the statement of the lemma, input sequences whose decoding process leaves a partial codeword $\mathtt{a}^i$ for $1 \le i \le k-2$ produce the same value as if that partial codeword were omitted.[4] ◀

3.2 Proof of Theorem 1c

Lemma 6 blows up the number of states by a factor that depends on the number of matrices. Thus, it is advantageous to apply it *after* merging the start matrix into π, when the number of matrices is reduced.

So we start with the 5-pair instances of Neary [21] and construct five matrices $A_i = A(v_i^R, w_i^R)$ from the reversed pairs. We then combine A_5 with the starting distribution π_1 into $\pi_2 = \pi_1 A_5$, leaving $k = 4$ matrices A_1, A_2, A_3, A_4 of dimension $d = 6$.

We cannot apply the transformations of Step 2, because it changes the first row, and state 1 would no longer be absorbing, which precludes the application of Lemma 6.

Thus, we apply Lemma 6 to the matrices A_1, A_2, A_3, A_4, resulting in two matrices $M'_{\mathtt{a}}$ and $M'_{\mathtt{b}}$ of dimension $(k-1)(d-1)+1 = 16$. The new start vector π_3 is π_2 padded with zeros, and the end vector f is still the first unit vector (of dimension 16).

We would now like to use the transformation of Step 2 to make the matrices column-stochastic. However, since we have replaced the initial start vector π_1 by π_2, the entries of the start vector after the transformation would not longer sum to 0, a property that is crucial for eventually making the matrices stochastic in Step 4.

Therefore we have to achieve column sums 1 in a different way, with an adaptation of the method of Step 3 (Section 2.5), adding an extra state;

---

4 Hirvensalo [15, p. 314] defined the vector f' ($\mathbf{y}_3$ in his notation) "analogously" to the vector π' ($\mathbf{x}_3$ in his notation), thus padding it with zeros. We see that with this definition, the result for incomplete inputs is $x_1 f_1$, which, in the general setting of our lemma, it has no predictable relation with the meaningful value $x_1 f_1 + \hat{x}\hat{f}$. In Hirsensalo's case, $x_1 f_1$ can be shown by a delicate argument to be ≤ 0, see [24, Section 8.2.3, footnote 31] in the updated version on the author's homepage.

3.3 Step 2′: Making column sums 1 with an extra state

We add an extra row s_i and an extra column to each matrix, ensuring that all column sums become 1.

$$B_1 = \begin{pmatrix} M_{\mathtt{a}} & 0 \\ s_1 & 1 \end{pmatrix}, \; B_2 = \begin{pmatrix} M_{\mathtt{b}} & 0 \\ s_2 & 1 \end{pmatrix} \tag{14}$$

We now have two 17×17 matrices B_1, B_2 with column sums 1. The remaining steps (Steps 3–5) are as before, with the appropriate adaptations, and they add another row and column. (We may have to choose a smaller constant α in Step 4.) This concludes the proof of Theorem 1c. ◀

We mention that the combined effect of Steps 2′–5 can also be achieved by applying the sharpened version of Turakainen's Theorem [28, Theorem 1(i)] from 1975, which converts any generalized PFA into a PFA with just 2 more states. (This method could already have been used by Blondel and Canterini in 2003 and by Hirvensalo in 2007, who used 4 extra states to achieve the same effect, see [2, Steps 3 and 4] and [15, Steps 4 and 5].) Since we had almost all tools available, we have chosen to describe the conversion directly. In fact, our procedure more or less parallels Turakainen's proof, except that we do not have to treat the vector f because it is already a 0-1-vector (Step 1 of [28]). Steps 2′ and 3 are usually treated together as one step. Our Step 4, the conversion to positive matrices, is more streamlined than in other proofs in the literature.[5]

3.4 Proof of Theorem 1d (two fixed matrices)

This is obtained by adapting the Proof of Theorem 1c in the same way as for Theorem 1b (Section 2.10). Instead of Neary's 5-pair instances, we take the instance $(v_1, w_1), \dots, (v_7, w_7)$ of Matiyasevich and Sénizergues and construct seven matrices $A_i = A(v_i, w_i)$ from these pairs (without reversing the words v_i and w_i). We then combine A_1, which must be the first matrix in the product, with the starting distribution π_1 into $\pi_2 = \pi_1 A_1$. The remaining matrices $A_2, A_3, \dots, A_7$ are fixed. Only π_2 is an input to the problem.

The remainder of the proof is the same as in Theorem 1c. We apply Lemma 6 to $k = 6$ matrices A_i of dimension $d = 6$, resulting in two fixed matrices $M'_{\mathtt{a}}$ and $M'_{\mathtt{b}}$ of dimension $(k-1)(d-1)+1 = 26$. The conversion to a PFA requires two more states, and this leads to Theorem 1d.[6] ◀

4 Proof of Theorem 2 (variable end vector f)

One can relax the requirement that f is a 0-1-vector and allow arbitrary values. If the values are in the interval $[0, 1]$ we can think of the entry f_i as a probability in a final acceptance decision, if the PFA is in state i after the input has been read. Another possibility is that f_i represents a *prize* or *reward* that that is gained when the process stops in state i. Then f_i

---

[5] We take this occasion to point out a slight mistake in the statement and proof of [28, Theorem 1(i)]. The cutpoint is not $1/(n+2)$ but it must be modified by the quantity d from [28, top of p. 30]. For the case $\pi f > 0$, the original statement of [28, Theorem 1(i)] can still be recovered from the stronger statement of [28, Theorem 1(iv)] by rescaling.

[6] We mention that *one* of the two matrices in Theorem 1c can also be held fixed: Neary's PCP contains one fixed word pair $(v_4, w_4) = (\mathtt{1}, \mathtt{0})$ [21, Theorem 11]. We see that the matrix $M'_{\mathtt{a}}$ in the construction of Lemma 6 contains only one of the matrices M_i of the original k-state automaton. (In the example shown, it is the matrix M_4.) If we arrange that this is the matrix constructed from the fixed pair (v_4, w_4), then $M'_{\mathtt{a}}$, and hence the corresponding stochastic 18×18 matrix in the final PFA, will be fixed.

does not need to be restricted to the interval $[0,1]$. In this view, instead of the acceptance probability, we compute the *expected* reward of the automaton, as in game theory. We call f_q the *output values*, and f and the *output vector* or the *end vector*, in analogy to the start vector π.

Since the transition matrices B_i after Step 2 are column-stochastic and positive by Lemma 5, we transpose them to produce stochastic matrices, which can be directly used as transition matrices. This will reverse the order of the matrices in the product and swap the start vector with the end vector:

$$\begin{aligned}\pi_2 B_{i_1} B_{i_2} \dots B_{i_m} f_2 &= f_2^T B_{i_m}^T B_{i_{m-1}}^T \dots B_{i_1}^T \pi_2^T \\ &= \pi_6 B_{i_m}^T B_{i_{m-1}}^T \dots B_{i_1}^T f_6 \qquad (15)\end{aligned}$$

We may prefer to have the end vector f in the interval $[0,1]$. Thus, we replace $f_6 = \pi_2^T$ by $f_7 = \big((2,2,2,2,2,2) + \pi_2\big)^T/12$, in analogy to Section 2.7. This vector is positive and has sum 1, and in particular, the entries lie between 0 and 1. The effect is to increase the value of (15) by 1/6.

We take the 5-pair instances of Neary [21] (Section 2.8) and construct five matrices $A_i = A(v_i^R, w_i^R)$ from the reversed pairs. The words v_i and w_i in these instances are nonempty, as required by Lemma 5 [21, Proof of Theorem 11]. We convert them to column-stochastic matrices B_i and use the transposed matrices B_i^T. We know that (v_5^R, w_5^R) must be used at the beginning of the PCP solution $i_1 i_2 \dots i_m$ ($i_1 = 5$) and nowhere else. Hence B_5^T must be used at the end of the product in (15), and we can merge it with f_6 into an end vector $f_8 = B_5^T f_7 = B_5^T \pi_2^T$. The four remaining matrices $B_1^T, B_2^T, B_3^T, B_4^T$ form the set $\mathcal{M}$. The starting distribution $\pi_6 = f_2^T$ is a unit vector, i.e., there is a single deterministic start state. This proves Theorem 2a for $\lambda = 1/6$.

4.1 Changing the cutpoint λ by manipulating the end vector f

When the end vector f of a PFA is under control, one can change the cutpoint λ to any desired value in the open interval between 0 and 1: Adding a constant K to all output values f_i will increase the expected output value by K, and scaling by a positive constant will affect the expected output value in the same way.

Thus, by changing all f_i to αf_i for $0 < \alpha < 1$, one may decrease λ to any positive value. By changing all f_i to $1 - \alpha + \alpha f_i$ for $0 < \alpha < 1$, one may increase λ to any value less than 1. If f is not constrained to the interval $[0,1]$, one can reach any real value λ.

4.2 Proof of Theorem 2b (fixed transition matrices)

We would like to apply the approach of Theorem 2a to the the seven word pairs (v_1, w_1), $\dots$, (v_7, w_7) from Section 2.10. They should fulfill the assumption of Lemma 5 that none of the words is empty. However, one of the rules of the 3-rule semi-Thue system of Matiyasevich and Sénizergues, the rule $x\bar{x} \longrightarrow \epsilon$, contains an empty word, see [19, System S_5, (23)]. The reduction of Claus [5] (see also [13, Theorem 4]) would translate this into a PCP pair with an empty word.

Luckily, this reduction can be patched to yield seven pairs of nonempty words v_i and w_i. We refer to Appendix B for details.

As above, we form from these pairs the stochastic matrices $B_1^T, \dots, B_7^T$. Only B_1^T is a variable matrix, and all other matrices are fixed. In the product $\pi_6 B_{i_m}^T B_{i_{m-1}}^T \dots B_{i_1}^T f_7$, we merge the first matrix with π_6 into $\pi_8 := \pi_6 B_{i_m}^T = \pi_6 B_2^T$, which becomes the fixed start vector, and the last matrix into $f_8 := B_{i_1}^T f_7 = B_1^T f_7$, which becomes the only input to the problem. The remaining five matrices $B_3^T, B_4^T, B_5^T, B_6^T, B_7^T$ form the fixed set $\mathcal{M}$. This proves Theorem 2b for $\lambda = 1/6$, and as above, it can be changed to any cutpoint λ. ◀

5 Outlook

The natural question is to ask for the smallest number of states for which PFA Emptiness is undecidable. Even if we consider generalized PFAs (rational-weighted automata), we could not reduce this number below 6. Claus [6, Theorem 7 and Corollary, p. 155] showed that the emptiness question can be decided for PFAs with two states.

A matrix dimension of six seems to be the minimum required in order to carry enough information to model concatenation of word pairs and allow testing for equality by a quadratic expression like (5), even in weighted automata. Undecidability for five or perhaps even three states would require some completely different (perhaps geometric) approach.

It would be an interesting exercise to trace back the undecidability proof of Matiyasevich and Sénizergues [19] to its origins and explicitly work out the fixed matrices of Theorem 1b or 2b. For one of the weaker results mentioned after Theorem 2, [24, Theorem 4b], one particular matrix from a set of 52 fixed 11×11 matrices is shown in [24, Section 7.3].

PFAs can be have other merits than just a small number of states and input symbols. We discuss some of these criteria.

Positive and doubly-stochastic transition matrices. In our results, the transition matrices can always be assumed to be strictly positive and sometimes also doubly-stochastic. Whenever this is the case, we have mentioned it in Theorems 1 and 2.

5.1 Obtaining a substantial gap between acceptance and rejection

As seen in formula (5), the acceptance probability barely rises above the threshold 0 when the input represents a solution of the PCP. (This tiny gap is further reduced by multiplying it with α^m in Step 4.) Thus, our constructions depend on the capability of a PFA to "detect" minute fluctuations of the acceptance probability above the cutpoint. This statement applies to all undecidability proofs in the Nasu–Honda line of descent.

By contrast, the proof of Condon and Lipton [7] from 1989 gives a more robust result, see also [24, Section 4]: For any $\varepsilon > 0$, it yields a PFA such that the largest acceptance probability is either $\geq 1 - \varepsilon$ or $\leq \varepsilon$, and the problem to detect which of the two cases holds is undecidable. Undecidability is derived from the halting problem for 2-counter machines, and the number of states of the PFA is beyond control.

Luckily, our results can be strengthened to even surpass the strong separation achieved in the Condon–Lipton construction, by the following gap amplification technique of Gimbert and Oualhadj, which we formulate in slightly generalized form.

▶ **Theorem 7** (Gimbert and Oualhadj 2010 [11]). *We are given a PFA A with input alphabet Σ and d states, with an output vector $f \in [0,1]^d$. We denote its acceptance probability for an input $u \in \Sigma^*$ by $\phi(u)$. Let λ_A, λ_B be two thresholds with $0 \leq \lambda_A \leq 1$ and $0 < \lambda_B < 1$.*

Then, from the description of A, we can construct a PFA B with input alphabet $\Sigma \cup \{\mathtt{end}, \mathtt{check}\}$ with $2d+3$ states, a single start state and a single accepting state, with the following property.

1. *If every input $u \in \Sigma^*$ for A has acceptance probability $\phi(u) \leq \lambda_A$, then every input for B is accepted by B with probability $\leq \lambda_B$.*

2. *If A has an input $u \in \Sigma^*$ with acceptance probability $\phi(u) > \lambda_A$, then B has inputs with acceptance probability arbitrarily close to* 1.

This was proved for $\lambda_A = \lambda_B = 1/2$ by Gimbert and Oualhadj [11] in 2010 in order to show that it is undecidable whether the conclusion of Case 2 for a PFA B holds (the "Value 1 Problem"). The construction and proof was simplified by Fijalkow as part of a short 8-page note [9]. The generalization to arbitrary λ_A and λ_B is not difficult, and in addition, we have included the precise statement about the number $2d + 3$ of states. For completeness, we present a proof in Appendix A. It essentially follows Fijalkow's construction, and it eliminates an oversight in [11] and [9].

With this technique, one can achieve an arbitrarily large gap with a moderate increase in states, roughly by a factor of 2. In particular, from Theorem 2a, we get a PFA B with 6 matrices of size 15×15, which exhibits the strong dichotomy expressed in Theorem 7, for any $\lambda_B > 0$.

This construction does not preserve the property of being a PFA with fixed transition matrices. In the PFA B constructed in Appendix A (see Table 2), the transitions depend both on the starting distribution π and on the final vector f of A.

5.2 Uniqueness

In Theorem 1a and Theorem 2a, the constructed PFA has the property that the recognized language contains at most one word. As with the large gap guarantee in Section 5.1, this leads to a stronger statement where the problem gets the nature of a *promise problem*.

Neary [21] derived his undecidable PCP instances from *binary tag systems*. A binary tag system performs a deterministic computation. It follows from the correctness argument of the simulation that the PCP solution is unique if it exists, apart from the obvious possibility of repeating the solution arbitrarily. In Section 2.8, we have excluded the last possibility by fixing the end pair (u_5^R, w_5^R) to be the first pair and removing it from the list. Thus, uniqueness holds for the PFA in Theorems 1a and 2a.

However, in the conversion to a binary input alphabet for Theorem 1c (Section 3.1), uniqueness is lost: We have seen that we may always add `a` or `aa` to a solution. We don't see an easy way to eliminate these extra solutions without increasing the number of states.

For Theorems 1b (Section 2.10) and 2b (Section 4.2), we used a PCP that models semi-Thue systems. However, even some 3 rule semi-Thue systems, like the one of [19], would have at most one derivation (which we have not checked), uniqueness would not survive the reduction to the PCP, see the remark in Appendix B, Step 1, after Claim 9. Thus we cannot claim uniqueness in these cases.

5.3 Simple probabilistic automata

In a *simple* PFA, all probabilities are 0, $\frac{1}{2}$, or 1 [11, Definition 2]. We have constructed our PFA with decimal fractions, but it would not be hard to switch to binary fractions. Before Step 4, the number of states should be increased to a power of two, so that the entries of J become binary fractions as well. Once all transition probabilities are binary fractions, the PFA can be converted to a simple PFA by padding each input symbol with sufficiently many dummy symbols so that the PFA has time to make its random decisions with a sequence of unbiased coin flips; see [24, Section 4.4, proof of Theorem 1, item (b)] Thus the results can be achieved with simple PFAs, but with a larger number of matrices and a larger number of states. The precise quantities would depend on the lengths of the words v_i and w_i in the PCP.

References

1. Jean Berstel, Dominique Perrin, and Christophe Reutenauer. *Codes and Automata*, volume 129 of *Encyclopedia of Mathematics and its Applications*. Cambridge University Press, 2009.
2. Vincent D. Blondel and Vincent Canterini. Undecidable problems for probabilistic automata of fixed dimension. *Theory Comput. Systems*, 36:231–245, 2003. doi:10.1007/s00224-003-1061-2.
3. Vincent D. Blondel and John N. Tsitsiklis. The boundedness of all products of a pair of matrices is undecidable. *Systems & Control Letters*, 41(2):135–140, 2000. doi:10.1016/S0167-6911(00)00049-9.
4. Vuong Bui. *Growth of Bilinear Maps*. PhD thesis, Freie Universität Berlin, Institut für Informatik, 2023. doi:10.17169/refubium-41705.
5. Volker Claus. Some remarks on PCP(k) and related problems. *Bull. Europ. Assoc. Theoret. Computer Sci. (EATCS)*, 12:54–61, October 1980. URL: http://page.mi.fu-berlin.de/rote/Kram/Volker_Claus-Some_remarks_on_PCP(k)_and_related_problems-1980-Bull-EATCS-12.pdf.
6. Volker Claus. The (n, k)-bounded emptiness-problem for probabilistic acceptors and related problems. *Acta Informatica*, 16:139–160, 1981. doi:10.1007/BF00261257.
7. A. Condon and R. J. Lipton. On the complexity of space bounded interactive proofs. In *Proceedings of the 30th Annual Symposium on Foundations of Computer Science*, SFCS '89, pages 462–467, USA, 1989. IEEE Computer Society. doi:10.1109/SFCS.1989.63519.
8. Ruiwen Dong. Recent advances in algorithmic problems for semigroups. *ACM SIGLOG News*, 10(4):3–23, December 2023. doi:10.1145/3636362.3636365.
9. Nathanaël Fijalkow. Undecidability results for probabilistic automata. *ACM SIGLOG News*, 4(4):10–17, November 2017. doi:10.1145/3157831.3157833.
10. Rūsiņš Freivalds. Probabilistic two-way machines. In Jozef Gruska and Michal Chytil, editors, *Mathematical Foundations of Computer Science 1981 (MFCS)*, volume 118 of *Lecture Notes in Computer Science*, pages 33–45. Springer, 1981. doi:10.1007/3-540-10856-4_72.
11. Hugo Gimbert and Youssouf Oualhadj. Probabilistic automata on finite words: Decidable and undecidable problems. In Samson Abramsky, Cyril Gavoille, Claude Kirchner, Friedhelm Meyer auf der Heide, and Paul G. Spirakis, editors, *Automata, Languages and Programming. 37th International Colloquium, ICALP 2010, Bordeaux, France, July 2010, Proceedings, Part II*, volume 6199 of *Lecture Notes in Computer Science*, pages 527–538, Berlin, Heidelberg, 2010. Springer-Verlag. Full version in https://hal.science/hal-00456538v3. doi:10.1007/978-3-642-14162-1_44.
12. Vesa Halava and Tero Harju. On Markov's undecidability theorem for integer matrices. *Semigroup Forum*, 75:173–180, 2007. doi:10.1007/s00233-007-0714-x.
13. Vesa Halava, Tero Harju, and Mika Hirvensalo. Undecidability bounds for integer matrices using Claus instances. *International Journal of Foundations of Computer Science*, 18(05):931–948, 2007. doi:10.1142/S0129054107005066.
14. Tero Harju and Juhani Karhumäki. Morphisms. In Grzegorz Rozenberg and Arto Salomaa, editors, *Handbook of Formal Languages: Volume 1. Word, Language, Grammar*, chapter 7, pages 439–510. Springer, Berlin, Heidelberg, 1997. doi:10.1007/978-3-642-59136-5_7.
15. Mika Hirvensalo. Improved undecidability results on the emptiness problem of probabilistic and quantum cut-point languages. In Jan van Leeuwen, Giuseppe F. Italiano, Wiebe van der Hoek, Christoph Meinel, Harald Sack, and František Plášil, editors, *SOFSEM 2007: Theory and Practice of Computer Science*, volume 4362 of *Lecture Notes in Computer Science*, pages 309–319, Berlin, Heidelberg, 2007. Springer. doi:10.1007/978-3-540-69507-3_25.
16. John E. Hopcroft and Jeffrey E. Ullman. *Introduction to Automata Theory, Languages and Computation*. Addison-Wesley, 1979.
17. Omid Madani, Steve Hanks, and Anne Condon. On the undecidability of probabilistic planning and related stochastic optimization problems. *Artif. Intell.*, 147(1–2):5–34, 2003. doi:10.1016/S0004-3702(02)00378-8.

18 A. Markov. On certain insoluble problems concerning matrices. *Doklady Akad. Nauk SSSR (N.S.)*, 57:539–542, 1947. (Russian).

19 Yuri Matiyasevich and Géraud Sénizergues. Decision problems for semi-Thue systems with a few rules. *Theoretical Computer Science*, 330(1):145–169, 2005. `doi:10.1016/j.tcs.2004.09.016`.

20 Masakazu Nasu and Namio Honda. Mappings induced by PGSM-mappings and some recursively unsolvable problems of finite probabilistic automata. *Information and Control*, 15(3):250–273, 1969. `doi:10.1016/S0019-9958(69)90449-5`.

21 Turlough Neary. Undecidability in binary tag systems and the Post correspondence problem for five pairs of words. In Ernst W. Mayr and Nicolas Ollinger, editors, *32nd International Symposium on Theoretical Aspects of Computer Science (STACS 2015)*, volume 30 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 649–661, Dagstuhl, Germany, 2015. Schloss Dagstuhl–Leibniz-Zentrum für Informatik. `doi:10.4230/LIPIcs.STACS.2015.649`.

22 Michael S. Paterson. Unsolvability in 3×3 matrices. *Stud. in Appl. Math.*, 49(1):105–107, 1970. `doi:10.1002/sapm1970491105`.

23 Azaria Paz. *Introduction to Probabilistic Automata*. Computer Science and Applied Mathematics. Academic Press, New York, 1971. `doi:10.1016/c2013-0-11297-4`.

24 Günter Rote. Probabilistic Finite Automaton Emptiness is undecidable, June 2024. URL: `http://page.mi.fu-berlin.de/rote/Papers/pdf/Probabilistic+Finite+Automaton+Emptiness+is+undecidable.pdf`, `arXiv:2405.03035`.

25 Günter Rote. Probabilistic Finite Automaton Emptiness is undecidable for a fixed automaton. In Paweł Gawrychowski, Filip Mazowiecki, and Michał Skrzypczak, editors, *50th International Symposium on Mathematical Foundations of Computer Science (MFCS 2025)*, volume 345 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 86:1–86:18. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2025. `arXiv:2412.05198`, `doi:10.4230/LIPIcs.MFCS.2025.86`.

26 Michael Sipser. *Introduction to the Theory of Computation*. PWS Publishing Company, 1997.

27 Paavo Turakainen. Generalized automata and stochastic languages. *Proc. Amer. Math. Soc.*, 21:303–309, 1969. `doi:10.1090/S0002-9939-1969-0242596-1`.

28 Paavo Turakainen. Word-functions of stochastic and pseudo stochastic automata. *Annales Fennici Mathematici*, 1(1):27–37, February 1975. URL: `https://afm.journal.fi/article/view/134240`, `doi:10.5186/aasfm.1975.0126`.

A Proof of Theorem 7 (gap amplification)

▶ **Theorem 7** (Gimbert and Oualhadj 2010 [11]). *We are given a PFA A with input alphabet Σ and d states, with an output vector $f \in [0,1]^d$. We denote its acceptance probability for an input $u \in \Sigma^*$ by $\phi(u)$. Let λ_A, λ_B be two thresholds with $0 \le \lambda_A \le 1$ and $0 < \lambda_B < 1$.*

Then, from the description of A, we can construct a PFA B with input alphabet $\Sigma \cup \{\texttt{end}, \texttt{check}\}$ with $2d+3$ states, a single start state and a single accepting state, with the following property.

1. *If every input $u \in \Sigma^*$ for A has acceptance probability $\phi(u) \le \lambda_A$, then every input for B is accepted by B with probability $\le \lambda_B$.*
2. *If A has an input $u \in \Sigma^*$ with acceptance probability $\phi(u) > \lambda_A$, then B has inputs with acceptance probability arbitrarily close to 1.*

Proof. We first describe B and explain its operation under the assumptions that $\lambda_A = 1/2$ and f is a 0-1-vector; in other words, A has a set F of accepting states and a complementary set R of rejecting states.

The input for the PFA B is structured into *rounds*. Each round is of the form

$$\ldots \texttt{check}\ u_1\ \texttt{end}\ u_2\ \texttt{end}\ \ldots\ \texttt{end}\ u_n\ \texttt{end}\ u_{n+1}\ \texttt{check} \ldots \tag{16}$$

with $u_i \in \Sigma^*$. The check symbols delimit each round and separate it from the adjacent rounds. The end symbols separate successive strings u_i, which are taken as inputs for (a simulation of) A. The last string u_{n+1} is ignored: It has no effect on the outcome.

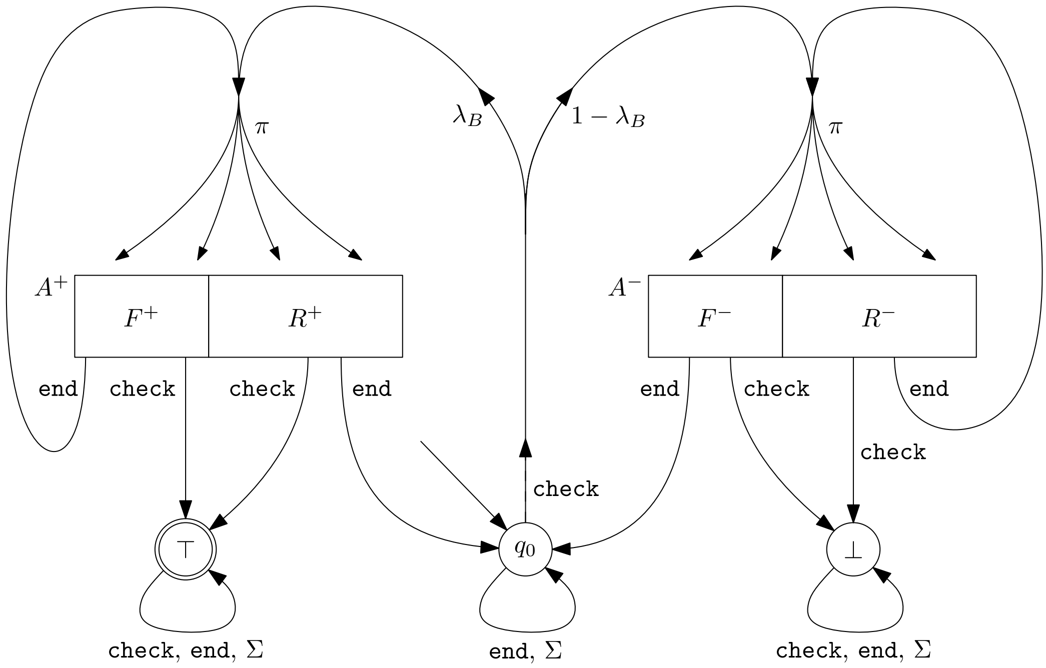

Figure 1 The expanding PFA B

The PFA B contains two copies A^+ and A^- of the machine A. There are three additional states: a start state q_0 and two absorbing states $\top$ and $\bot$. The illustration in Figure 1 shows each copy of A split into accepting states F and rejecting states R. Each bundle of arrows leading into A^+ and A^- indicates a transition according to the input distribution π of A.

The machine B behaves as follows: At the beginning of each round, is is either in one of the two absorbing states or in state q_0. If it is in state q_0, it flips a biased coin, and with probabilities λ_B and $1-\lambda_B$, it does one of the following two things:

(+) If *all* strings $u_1, \dots, u_n$ of this round were accepted by A, the machine B accepts the input, moving to the absorbing state $\top$. Otherwise, it moves to q_0 and waits for the next round.

(−) If *all* strings $u_1, \dots, u_n$ of this round were rejected by A, the machine B rejects the input, moving to the absorbing state $\bot$. Otherwise, it moves to q_0 and waits for the next round.

We call the three possible outcomes of a round ACCEPT, REJECT, and INDECISION. If the PFA never takes a decision before the input is exhausted, it will end in some nonabsorbing state, and hence it will also reject the input.

It is easy to write down the probabilities of the outcomes of a round (16):

$$\Pr[\text{ACCEPT}] = \lambda_B \phi(u_1)\phi(u_2)\cdots\phi(u_n)\cdot \tag{17}$$

$$\Pr[\text{REJECT}] = (1-\lambda_B)\big(1-\phi(u_1)\big)\big(1-\phi(u_2)\big)\cdots\big(1-\phi(u_n)\big) \tag{18}$$

Case 1: If $\phi(u_i) \le 1/2$ for every u_i, then $\phi(u_i) \le 1-\phi(u_i)$, and it follows that

$$\frac{\Pr[\text{ACCEPT}]}{\Pr[\text{REJECT}]} \le \frac{\lambda_B}{1-\lambda_B},$$

and hence

$$\frac{\Pr[\text{ACCEPT}]}{\Pr[\text{decide}]} = \frac{\Pr[\text{ACCEPT}]}{\Pr[\text{ACCEPT}] + \Pr[\text{REJECT}]} \le \lambda_B,$$

For the purpose of the overall analysis, it helps to imagine that the machine *first* makes up its mind *whether* a decision is made, and only then decides *which* of the branches ACCEPT or REJECT is taken, as represented in Figure 2.

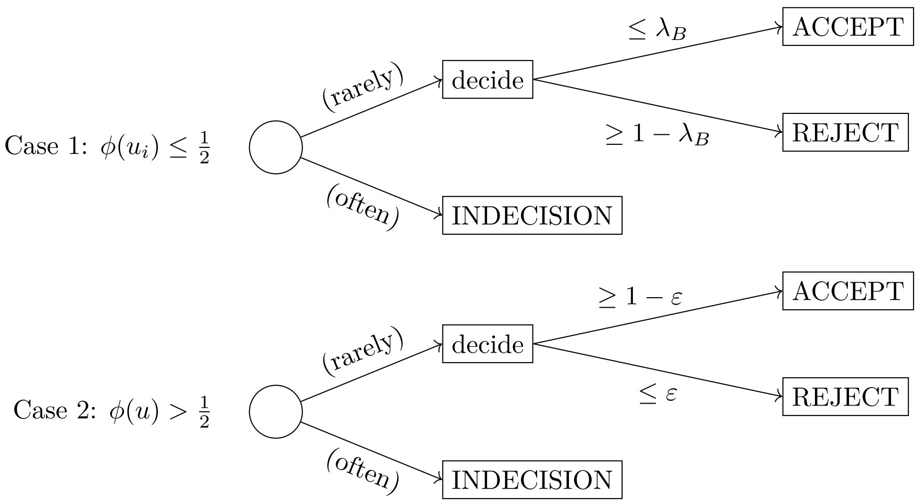

Figure 2 The outcome of processing one round by the PFA B. Case 2 assumes a large number n of repetitions of u.

Then, for an input that consists of any number of rounds, whenever the decision is taken, B favors REJECT over ACCEPT with a ratio at least $(1 - \lambda_B) : \lambda_B$. To this, we must add the probability of rejection due to continuing INDECISION, which is also in favor of rejection. This proves statement 1 of the theorem.

Case 2: We take a word u with $\phi(u) > 1/2$ and repeat it a large number of times. In this way, for a round of the form $\ldots\texttt{check}\ (u\ \texttt{end})^n\texttt{check}\ldots$, we can make the ratio between ACCEPT (17) and REJECT (18) as large as we like:

$$\frac{\Pr[\text{ACCEPT}]}{\Pr[\text{decide}]} = \frac{\Pr[\text{ACCEPT}]}{\Pr[\text{ACCEPT}] + \Pr[\text{REJECT}]} \ge 1 - \varepsilon, \tag{19}$$

for any $\varepsilon > 0$. The probability of INDECISION will also approach 1, but this can be compensated by a large number of rounds: We create an input of the following form:

$$\big(\texttt{check}\ (u\ \texttt{end})^n\big)^m$$

We first choose n large enough to satisfy (19) in each round. Then, no matter how tiny the probability of a decision in a single round is, the probability of continued INDECISION can be made arbitrarily small by increasing m. The input is rejected if either (a) no decision is taken or (b) the PFA takes a decision and it decides to REJECT. Both probabilities can be made arbitrarily small, and this proves statement 2 of the theorem.

We still have to discuss the cases that f is not a 0-1-vector or $\lambda_A \ne 1/2$. Consider a state i with $0 < f_i < 1$. If B is in such a state and the `end` symbol is read, then with probability f_i, it behaves like an accepting state, and with probability $1 - f_i$, it behaves like a rejecting state. This is reflected the transition table of Table 2.

state of B	input symbol	transition to	with probability
q_0	check	q_j^+ q_j^-	$\lambda_B \pi_j$ $(1-\lambda_B)\pi_j$
	end, Σ	q_0	1
q_i^+	end	q_j^+ q_0	$f_i \pi_j$ $1 - f_i$
	check	$\top$	1
	Σ	q_j^+	as from q_i to q_j in A
q_i^-	end	q_0 q_j^-	f_i $(1-f_i)\pi_j$
	check	$\bot$	1
	Σ	q_j^-	as from q_i to q_j in A

Table 2 Transitions in B for all states except the absorbing states $\top$ and $\bot$

Finally, if $\lambda_A \neq 1/2$, we have seen in Section 4.1 how we can modify f to get an equivalent PFA with $\lambda_A = 1/2$. Thus, the theorem is proved in full generality. ◀

B From a semi-Thue system with k rules to a PCP with $k+4$ pairs of nonempty words

The construction is due to Claus [5] from 1980, and it was refined by Halava, Harju, and Hirvensalo in 2007 [13, Theorem 6], with a condensed and slightly different proof. We give a variation of the proof, extending it to ensure that none of the words in the pairs is empty.

The following version of the PCP is suitable for our needs [14, Section 3.2]:

The Generalized Post Correspondence Problem (GPCP).
Given a sequence of pairs of words $(v_1, w_1), (v_2, w_2), \dots (v_k, w_k)$, is there a sequence $i_2, \dots, i_{m-1}$ of indices $i_j \in \{3, \dots, k\}$ such that

$$v_1 v_{i_2} v_{i_3} \dots v_{i_{m-1}} v_2 = w_1 w_{i_2} w_{i_3} \dots w_{i_{m-1}} w_2 \ ?$$

We have already encountered such special roles of the start pair (v_1, w_1) and the end pair (v_2, w_2), both in connection with the 5-pair PCPs of Neary and the 7-pair PCPs of Matiyasevich and Sénizergues. However, there it was a property of the PCP *instances*, of which this restricted form of the solutions was a *consequence*. In case of the GPCP, this is a *requirement* on the solution; it is part of the "rules of the game."

There is also the *Modified Post Correspondence Problem* (MPCP), where only the start pair (v_1, w_1) is prescribed. It is often used as an intermediate problem when reducing the Halting Problem for Turing machines to the PCP. This MPCP (or the GPCP) can be reduced to the classic PCP with some extra effort, by modifying the word pairs, see for example [16, Lemma 8.5] or [26, p. 189]. We describe this procedure, which follows the standard pattern, in Step 4 of the proof below.

In our situation, the GPCP is actually the more convenient version of the problem. Although we don't need the PCP version of the theorem in this paper, we include it for completeness, and also since the proof in [13, Theorem 6] contains a slight error, which makes is invalid in the very context of empty words.

The theorem starts from a *semi-Thue system.* A semi-Thue system is specified by a set of *replacement rules* of the form $l \longrightarrow r$, where l and r are words over some alphabet. Such a rule means that a word u can be transformed by replacing some occurrence of l in u by r. This is called a one-step derivation, and it is written as $u \to v$, meaning that $u = xly$ and $v = xry$ for some words x, y. As usual, $\xrightarrow{*}$ denotes the transitive and reflexive closure of this relation.

▶ **Theorem 8.** *Suppose there is a semi-Thue system with k rules $l_i \longrightarrow r_i$ over some alphabet Σ, and a word $u_\infty \in \Sigma^*$, for which the following problem (the* individual word problem *or* individual accessibility problem *with fixed target u_∞) is undecidable:*

Given a word $u_0 \in \Sigma^*$, is $u_0 \xrightarrow{*} u_\infty$?

Then there is a list of $2(k+4)-1$ nonempty words $v_1, v_2, v_3, \dots, v_{k+4}$ and $w_2, w_3, \dots, w_{k+4}$ over the binary alphabet $\{\mathtt{a}, \mathtt{b}\}$ for which the following problem is undecidable:

Given a nonempty word $w_1 \in \{\mathtt{a}, \mathtt{b}\}^*$, does the GPCP with the $k+4$ word pairs $(v_1, w_1), (v_2, w_2), \dots, (v_{k+4}, w_{k+4})$ have a solution?

The construction can be modified so that the same question with the (ordinary) PCP is undecidable.

Proof. We transform the semi-Thue system into a PCP in several steps.

Step 1: Representing a derivation as a GPCP solution. We start with a simpler procedure that creates $k+3+|\Sigma|$ word pairs. We expand the alphabet to the larger alphabet $\Sigma' := \Sigma \cup \{\#\}$ with a new separator symbol #. We use the following word pairs:

- the *start pair* $(v_1, w_1) = (\#, \#u_0\#)$.
- the *end pair* $(v_2, w_2) = (\#u_\infty\#, \#)$.

In addition, we have the following $k + |\Sigma| + 1$ word pairs:

- for every rule $l \longrightarrow r$, the *rule pair* (l, r)
- for every symbol $\sigma \in \Sigma'$, the *copying pair* (σ, σ)

The idea is that a derivation

$$u_0 \to u_1 \to \cdots \to u_{n-1} \to u_n = u_\infty \tag{20}$$

is represented by the PCP solution

$$v_1 v_{i_2} v_{i_3} \dots v_{i_{m-1}} v_2 = w_1 w_{i_2} w_{i_3} \dots w_{i_{m-1}} w_2 = \#u_0\#u_1\#\dots\#u_{n-1}\#u_\infty\#.$$

When building this solution from left to right, the partial solution $v_1 v_{i_2} v_{i_3} \dots v_{i_j}$ lags one derivation step behind the partial solution $w_1 w_{i_2} w_{i_3} \dots w_{i_j}$. After finishing each complete derivation step, the situation typically looks like this:

$$\begin{aligned} v_1 v_{i_2} v_{i_3} \dots v_{i_j} &= \#u_0\#u_1\#\dots\#u_{s-1}\# \\ w_1 w_{i_2} w_{i_3} \dots w_{i_j} &= \#u_0\#u_1\#\dots\#u_{s-1}\#u_s\# \end{aligned} \tag{21}$$

We will often use the image of building a solution incrementally from left to right in two rows as in (21). The words v_i are appended one after the other in the upper row as the corresponding words w_i are simultaneously concatenated in the lower row.

The start pair (v_1, w_1) establishes the situation (21) at the beginning. When extending this solution, the next word pair $(v_{i_{j+1}}, w_{i_{j+1}})$ and the following ones must match their v-components with the string u_s in the lower row, and they can do so either by simply using copying pairs, or by using the left side l of a rule pair (l, r), in which case r is substituted for l in the growing word u_{s+1} in the lower row. The solution can only be completed with the end pair $(\#u_\infty\#, \#)$, i.e., if $u_s = u_\infty$.

The following statements can be easily established by induction:

▷ Claim 9. **1.** At any time after the start pair (v_1, w_1) is included and before the last pair (v_2, w_2) is used, the partial solution $w_1 w_{i_2} w_{i_3} \dots w_{i_j}$ in the lower row contains one more symbol `#` than $v_1 v_{i_2} v_{i_3} \dots v_{i_j}$ in the upper row.

2. In particular, $v_1 v_{i_2} v_{i_3} \dots v_{i_j}$ is a prefix of $w_1 w_{i_2} w_{i_3} \dots w_{i_j}$ and strictly shorter than $w_1 w_{i_2} w_{i_3} \dots w_{i_j}$.

3. Two consecutive words u and u' that are delimited by `#` symbols are related in the following way: Any number of disjoint occurrences of the left hand side l of some rule $l \longrightarrow r$ in u are replaced by the corresponding right-hand side r. ◁

It is thus possible to carry out several disjoint replacements $l \longrightarrow r$, possibly with different rules, when creating u' from u, but these would be permitted in the semi-Thue system as a sequence of consecutive replacements. It follows that the GPCP has a solution if and only if $u_0 \xrightarrow{*} u_\infty$.

We see from this discussion that the successive words $u_0, u_1, \dots$ in the solution word (21) are not necessarily the same as the successive words $u_0, u_1, \dots$ in the derivation (20): They form a subsequence of (20), possibly with repetitions. We note in particular that uniqueness is lost in the transformation. Even if the semi-Thue system has a unique derivation (20), the GPCP will not have unique solutions. The reason is that the GPCP may decide, at any time, to use only copying pairs in one round, effectively setting $u_{s+1} = u_s$ in (21).

Step 2. Encoding the strings in binary. We choose a binary encoding $\Sigma' \to \{\mathtt{a}, \mathtt{b}\}$ of the alphabet, using the codewords `bab`, `baab`, `baaab`, `baaaab`, ... We denote the encoded version of a string w by $\langle w \rangle \in \{\mathtt{a}, \mathtt{b}\}^*$.

The advantage of the binary alphabet is that we need just two copying pairs. All other word pairs are simply copied symbol by symbol. Thus, we have the following $k + 4$ word pairs:

- the *start pair* $(v_1, w_1) = (\langle \# \rangle, \langle \# u_0 \# \rangle)$
- the *end pair* $(v_2, w_2) = (\langle \# u_\infty \# \rangle, \langle \# \rangle)$
- k *rule pairs* $(\langle l \rangle, \langle r \rangle)$, one for each rule $l \longrightarrow r$
- two *copying pairs* $(\mathtt{a}, \mathtt{a})$ and $(\mathtt{b}, \mathtt{b})$.

To ensure that this transformation is correct, one needs to use the fact that the family of codewords `bab`, `baab`, `baaab`, ... has some strong uniqueness properties. Not only is the code injective, but no codeword $\langle \alpha \rangle$ is a substring of another codeword or even of a composition of codewords, unless it appears there because it was used to encode the symbol α.[7] Such a code is sometimes called a *self-synchronizing code*.[8] Here is a formal statement of this property for the case of the binary alphabet $\{\mathtt{a}, \mathtt{b}\}$:

---

[7] Claus [5, p. 57, transformation from system U_1 to U_2] applies a code with three codewords $h(x_1) = \mathtt{01}$, $h(x_2) = \mathtt{011}$, $h(x_3) = \mathtt{0111}$ at this place of the argument (assuming that the initial semi-Thue system uses a binary alphabet, and by adding `#`, the alphabet has grown to three letters x_1, x_2, x_3). This code is injective, but it is inadequate and can lead to errors. For example, the first three bits of `0111` can be mistaken as the codeword `011` for x_2, and a replacement rule for a word ending in x_2 can be applied. The remaing bit `1` is simply copied by the copying pair. The rule might generate a word ending in x_1, and $h(x_1) = \mathtt{01}$ together with the remaining `1` forms the codeword `011`, which stands for x_2. So the derivation could proceed and the mistake would go undetected.
Other sources [12, 13] use codes with codewords $\mathtt{ab}^i\mathtt{a}$, which are essentially the same as our proposal, without mentioning their special properties.

[8] In [1, Eq. (7.12), p. 285], such codes are called *comma-free* codes. In other places, the term "*comma-free*" is reserved more specifically for self-synchronizing codes that are block-codes, i.e., where every codeword has the same length.

▶ **Proposition 10.** *If $\langle u\rangle = x\langle\alpha\rangle y$ for some symbol $\alpha \in \Sigma'$ and some $x, y \in \{\mathtt{a}, \mathtt{b}\}^*$, then there are strings $v, w \in (\Sigma')^*$ such that $\langle v\rangle = x$ and $\langle w\rangle = y$ (and consequently, $u = v\alpha w$).* ◀

Step 3: Making the rule pairs nonempty. When one of the sides l or r of a rule $l \longrightarrow r$ is empty, the corresponding word in the word pair $(\langle l\rangle, \langle r\rangle)$ is also empty. To avoid empty words, we change the rule pairs to $(\langle l\rangle\mathtt{b}, \langle r\rangle\mathtt{b})$ for each rule $l \longrightarrow r$.

The extra letter b makes no difference, because all codewords start with b. Thus, whenever we want to match $\langle l\rangle$ in the upper row, we are sure that the next symbol is b, so we might as well copy the b from the upper row and insert it after $\langle r\rangle$ on the lower row. We might lose an opportunity to carry out several rule replacements simultaneously, but this does not change the solvability of the GPCP. The only problematic case would be that $\langle l\rangle$ or $\langle r\rangle$ occurs at the end of the solution string, but this is impossible since the rules of the GPCP require the string to end with the last words $v_2 = \langle\#u_\infty\#\rangle$ and $w_2 = \langle\#\rangle$.

At this point, we have proved the theorem as far as the GPCP is concerned. We only need to observe that w_1 is the only word that depends on u_0, the input to the individual word problem for the semi-Thue system. All other words are fixed.

Optional Step 4: Forcing the start and end pairs to their places. For converting the GPCP to an ordinary PCP, we use the following construction of [14, Theorem 3.2]. We add three new symbols *, [, and] to the alphabet and define two morphisms $\sigma, \rho\colon \{\mathtt{a}, \mathtt{b}\}^* \to \{\mathtt{a}, \mathtt{b}, *\}^*$. The morphism $\sigma(u)$ adds a * on the left of every symbol of u, and $\rho(u)$ adds a * on the right of every symbol of u. For example, $\sigma(\mathtt{aab}) = $ *a*a*b and $\rho(\mathtt{aab}) = $ a*a*b*.

We define the modified word pairs $(\bar v_i, \bar w_i)$ as follows:

$$(\bar v_1, \bar w_1) = (\,[\sigma(v_1), [*\rho(w_1)) = (\,[\sigma(v_1), [\sigma(w_1)*)$$
$$(\bar v_2, \bar w_2) = (\sigma(v_2)*], \rho(w_2)]\,) = (*\rho(v_2)], \rho(w_2)]\,)$$
$$(\bar v_i, \bar w_i) = (\sigma(v_i), \rho(w_i)) \quad (i \ge 3)$$

The pattern of these words is shown in Table 3.

	$i = 1$	$3 \le i \le k+4$	$i = 2$
$\bar v_i$	$[*x*x*x*x$	$*x*x*x$	$*x*x*]$
$\bar w_i$	$[*x*x*$	$x*x*x*x*x*$	$x*]$

Table 3 Patterns of some representative words $\bar v_i$ and $\bar w_i$. Each symbol x stands for a or b. The lengths of these words can of course be different from what is shown: The number of x's is the length of the original words v_i or w_i.

It is clear that every solution to the GPCP gives rise to a solution of the PCP: For every initial part $1, i_2, \ldots, i_j$ of a PCP solution that does not include the final word pair $i_m = 2$, the following relation can be easily proved by induction.

$$\bar v_1 \bar v_{i_2} \bar v_{i_3} \ldots \bar v_{i_j} = [\,\sigma(v_1 v_{i_2} v_{i_3} \ldots v_{i_j}) \tag{22}$$
$$\bar w_1 \bar w_{i_2} \bar w_{i_3} \ldots \bar w_{i_j} = [\,\sigma(w_1 w_{i_2} w_{i_3} \ldots w_{i_j})\,* \tag{23}$$

When the solution includes the final pair $i_m = 2$, the extra * appears also at the end of (22), and the closing bracket sign] is added to both strings, so we have equality between (22) and (23).

For the reverse direction, we have to argue that a solution to the PCP for the word pairs $(\bar v_i, \bar w_i)$ gives rise to the a solution of the GPCP. Since in every modified pair $(\bar v_i, \bar w_i)$ except

$(\bar{v}_1, \bar{w}_1)$, the two words $\bar{v}_i$ and $\bar{w}_j$ start with different letters, the start pair $(\bar{v}_1, \bar{w}_1)$ is the only pair that can be used at the start of the PCP. (This argument relies on the fact that all words are nonempty.)

Moreover, the opening bracket `[` occurs only in $\bar{v}_1$ and $\bar{w}_1$. Therefore, if the pair $(\bar{v}_1, \bar{w}_1)$ is used a second time, these occurrences of $\bar{v}_1$ and $\bar{w}_1$ must be matched in a left-aligned position. We can then cut the solution $i_1 i_2 \ldots i_m$ at this point, and both pieces will be solutions of the PCP.[9] Thus, by cutting the solution before the first repeated use of the pair $(\bar{v}_1, \bar{w}_1)$ (if any), we can assume that we only consider a solution where $(\bar{v}_1, \bar{w}_1)$ is used at the beginning and nowhere else.

Looking at the last symbol of the solution, the symmetric argument establishes that $(\bar{v}_2, \bar{w}_2)$ is used at the end and nowhere else.

We have seen that every solution of the PCP with the word pairs $(\bar{v}_i, \bar{w}_i)$, possibly after cutting it before the second use of the pair $(\bar{v}_1, \bar{w}_1)$, must satisfy the constraints of the GPCP, and by (22–23), this gives a solution for the original GPCP. (This statement holds true also if the pairs may contain empty words, with the obvious exception of the pair (ϵ, ϵ), but this requires additional arguments.)

Step 5: Reduction to two characters. Step 4 has added three characters `[`, `]`, `*` to the alphabet. We can easily reduce {`a`, `b`, `[`, `]`, `*`} to a binary alphabet, for example with the fixed-length codewords `aaa`, `bbb`, `bba`, `aba`, and `bab`. This concludes the proof of Theorem 8.

B.1 Counterexample to Lemma 1 in Halava, Harju, and Hirvensalo [13]

Our proof approach differs slightly from the proof of Claus [5]: There, the transformation to the PCP (Step 1) is merged with the enforcement of the GPCP rules (Step 4) into one transformation. The binary coding (Step 2) is done *before* the transformation to the PCP (Step 1), already at the level of the semi-Thue system.

Halava, Harju, and Hirvensalo [13, Theorem 4] gave an independent presentation of Claus' construction.[10] Unfortunately, their proof mingles the different construction steps (Steps 1, 2, 4, and 5) into a single condensed one-shot construction, which thereby becomes less transparent. For the correctness, they refer to a lemma about PCP solutions, which, however, is subject to exceptions when empty words appear. (No proof of this lemma is given in [13]. The authors write that the lemma "is straightforward" and supply an unspecific reference to a 70-pages handbook article [14] of Harju and Karhumäki[11].)

It is easy to construct a counterexample to this lemma when one of the words is empty. In the setting of [13], a PCP with n word pairs (v_i, w_i) is specified by two morphisms $g, h\colon \{b_1, b_2, \ldots, b_n\} \to \Gamma^*$ such that $(h(b_i), g(b_i)) = (v_i, w_i)$. In line with [13], we translate the PCP into a binary alphabet with codewords of the form $\mathtt{ab}^{i+1}\mathtt{a}$, although this translation

[9] In the corresponding construction of [13, Lemma 1], the initial brackets `[` in $\bar{v}_1$ and $\bar{w}_1$ are omitted and the final brackets `]` in $\bar{v}_2$ and $\bar{w}_2$ are replaced by `*` symbols, which are denoted by d in [13]. The statement is then still true, but only for nonempty words v_i and w_i, see Section B.1. This version economizes in alphabet size, but correctness is more delicate.

[10] See also the TUCS Technical Report No. 766, Turku Centre for Computer Science, April 2006, by the same authors with the same title, `https://typeset.io/pdf/undecidability-bounds-for-integer-matrices-using-claus-1nwnni898j.pdf`

[11] In this handbook article, Harju and Karhumäki formulate Step 4 as a separate theorem [14, Theorem 3.2]. Still, they prove their version of Theorem 8 [14, Theorem 4.1] by a "refinement of the general ideas presented in the proof of Theorem 3.2", instead of simply *applying* that theorem.

is not important for our point. We use the following code $\varphi\colon \Gamma^* \to \{\mathtt{a},\mathtt{b}\}^*$: $\varphi(\mathtt{*}) = \mathtt{aba} =: d$, $\varphi(\mathtt{u}) = \mathtt{abba}$, $\varphi(\mathtt{v}) = \mathtt{abbba}$, $\varphi(\mathtt{w}) = \mathtt{abbbba}$.

The PCP of our example has three word pairs. So we use a 3-letter index alphabet $\{b_1, b_2, b_3 = b_n\}$, and the word pairs are:

$$\begin{array}{lll} g(b_1) = \varphi(\mathtt{*}) = d & h(b_1) = \varphi(\mathtt{*v*w}) & \text{(the start pair)} \\ g(b_2) = \varphi(\epsilon) = \epsilon & h(b_2) = \varphi(\mathtt{*u}) & \\ g(b_3) = \varphi(\mathtt{u*v*w**}) & h(b_3) = \varphi(\mathtt{**}) = dd & \text{(the end pair)} \end{array}$$

It can be checked that these pairs follow the pattern that is prescribed for what is called a *Claus instance* in [13]. Lemma 1 in [13] claims in particular that, for such an instance, all nonempty solutions must start with b_1.

However, $g(b_2b_1b_3) = h(b_2b_1b_3) = \varphi(\mathtt{*u*v*w**})$. So, $b_2b_1b_3$ is a solution that does not start with b_1. The reason for the failure is that, since $g(b_2)$ is empty, the argument that the solution cannot start with anything but b_1 is not valid.

The original proof of Claus [5] is correct in this point, since it is more generous with additional alphabet symbols. (Our extra symbol * is called β in the notation of [5]; the role of the bracket symbols [and] is played by $h(\gamma) = \mathtt{0111}$.)

To be fair, one should mention that [13] sketches an alternative correctness proof of the construction for their Theorem 4 (our Theorem 8), which is tailored to the PCP at hand and which is independent of Lemma 1. That proof, however, also glosses over the case of empty words.

C Program for checking the multiplicative law (Lemma 3) in sagemath

```
from sage.symbolic.function_factory import function

BASE = 10

sigma = function("sigma",nargs=1) # unevaluated function
def sigma_num(u): return float(int("0"+u,BASE)/BASE**len(u))
    # parameter u is a string; returns 0.u as a number

l=function("l",nargs=1) # unevaluated function
def l_num(u): return float(BASE**-len(u))  # 10^{-|u|}

var("v w")
A0 = matrix([ (1, 0, 0, 0, 0, 0),
    (sigma(v), l(v), 0, 0, 0, 0),
    (sigma(v)^2, 2*l(v)*sigma(v), l(v)^2, 0, 0, 0),
    (sigma(w), 0, 0, l(w), 0, 0),
    (sigma(w)^2, 0, 0, 2*l(w)*sigma(w), l(w)^2, 0),
    (sigma(v)*sigma(w), l(v)*sigma(w), 0, l(w)*sigma(v), 0, l(v)*l(w)) ])

print("A0(v,w)="); print(A0,"\n")

def A0_numeric(vv,ww): return A0.subs(
        [l(v) == l_num(vv), sigma(v) == sigma_num(vv),
         l(w) == l_num(ww), sigma(w) == sigma_num(ww)] )
    # substitution of numerical values

v1,w1="12","999"
v2,w2="654","77"
print("A0("+v1+","+w1+")="); print(A0_numeric(v1,w1),"\n")

print("test multiplicative law, up to machine precision:")
diff = A0_numeric(v1,w1)*A0_numeric(v2,w2)-A0_numeric(v1+v2,w1+w2)
print("max deviation:", max(map(abs,diff.list())))
```

This script can be directly fed as input to sage[12]. It is robust against the loss of proper indentation.

[12] https://www.sagemath.org/