

Probabilistic Finite Automaton Emptiness is Undecidable

Günter Rote*

July 1, 2025

Abstract

It is undecidable whether the language recognized by a probabilistic finite automaton is empty. Several other undecidability results, in particular regarding problems about matrix products, are based on this important theorem. We present three proofs of this theorem from the literature in a self-contained way, and we derive some strengthenings. For example, we show that the problem remains undecidable for a fixed probabilistic finite automaton with 11 states, where only the starting distribution is given as input.

Contents

1	Probabilistic finite automata (PFA)	3
1.1	Formal problem definition	3
2	Statement of results	4
3	Preface: history and matrix products	7
3.1	Three proofs	7
3.2	Interlude: Other problems on matrix products	7
3.3	... back to the proofs of PFA Emptiness:	8
3.4	Overview	8
3.5	Comparison of the proofs	9
4	The Condon–Lipton proof via 2-counter machines	9
4.1	The Equality Checker	10
4.2	Correctness Test: checking a 2CM computation	12
4.3	Third-level aggregation: processing the whole input	13
4.3.1	Increasing the acceptance probability	13
4.3.2	Who is afraid of small probabilities?	14
4.3.3	Boosting the decision probabilities	14
4.4	Summing up the proof of Theorem 1	14
4.5	History of ideas	15
5	The Nasu–Honda–Claus proof via Post’s Correspondence Problem	15
5.1	The binary PFA	16
5.2	Post’s Correspondence Problem (PCP)	16
5.3	Testing equality of probabilities	17
5.4	Achieving strict inequality	18
5.5	History of ideas	19
5.5.1	Gimbert and Oualhadj 2010, following Bertoni 1975	19
5.6	Saving two states by merging indistinguishable states	21
5.7	Saving the start state by using the Modified Post Correspondence Problem	22

*Freie Universität Berlin, Institut für Informatik, Takustr. 9, 14195 Berlin, rote@inf.fu-berlin.de

C41	6 Fixing the set of matrices by using a universal Turing machine	23
C42	6.1 Constructing an MPCP for a Turing machine	24
C43	6.2 List of word pairs of the MPCP	25
C44	6.3 Using a universal Turing machine	26
C45	6.4 An efficient code	27
C46	6.5 Example matrices	28
C47	7 Output values instead of a set of accepting states	29
C48	7.1 Saving one more state by maintaining four binary variables	29
C49	7.2 Making all transition probabilities positive	30
C50	7.3 Fixing everything except the output vector, proof of Theorem 4	31
C51	7.4 Uniqueness of the solution	32
C52	7.5 Eliminating the output vector, proof of Theorem 2	33
C53	7.6 Reduction to 2 input symbols, proof of Theorem 3	35
C54	8 Using integer matrices	37
C55	8.1 The smallest number of states without regard to the size of the alphabet,	
C56	Theorem 5	37
C57	8.1.1 Step 5.1. Modeling the PCP by integer matrices	37
C58	8.1.2 Step 5.2. Introducing a unit vector as an ending vector	37
C59	8.1.3 Step 5.3. Conversion to stochastic matrices	38
C60	8.2 Binary alphabet, Theorem 6	39
C61	8.2.1 Step 6.1. Modeling the PCP by integer matrices	39
C62	8.2.2 Step 6.2. Merging the first and last matrices into the boundary	
C63	vectors	40
C64	8.2.3 Step 6.3. Reduction to two matrices	40
C65	8.2.4 Step 6.4. Introducing unit vectors as starting and ending vectors .	42
C66	8.2.5 Step 6.5. Conversion to stochastic matrices	42
C67	8.3 History of ideas	42
C68	9 Alternative universal Turing machines	43
C69	9.1 Watanabe, weak and semi-weak universality	43
C70	9.2 Wolfram–Cook, rule 110	43
C71	9.3 Wolfram’s 2,3 Turing machine	45
C72	10 Outlook	45
C73	10.1 Equality testing	45
C74	10.2 Shortcutting the reduction?	46
C75	10.3 The minimum number of states	46
C76	11 Epilogue: How to present a proof	47
C77	11.1 Levels of abstraction	47
C78	11.2 Case study 1: Restricting the output vector f to a 0-1-vector	47
C79	11.3 Case study 2: Probability amplification	48
C80	11.3.1 Algorithm F, Fijalkow	48
C81	11.3.2 Algorithm GO, Gimbert and Oualhadj	50
C82	11.3.3 The no-coin algorithm NC	51
C83	11.3.4 High-level verbal description versus state diagrams	52
C84	11.4 Case study 3: Coding in binary	54
C85	11.4.1 Integer matrices	55
C86	11.5 Case study 4: Testing equality	56
C87	11.6 Using auxiliary results or starting from scratch	57

C88	References	57
C89	A The original Nasu–Honda proof in a nutshell	61
C90	A.1 Deciding whether the recognized language is a regular language, or whether	
C91	it is context-free	63
C92	B Notation, terminology, and abbreviations	64

C93 1 Probabilistic finite automata (PFA)

C94 A probabilistic finite automaton (PFA) combines characteristics of a finite automaton
C95 and a Markov chain. We give a formal definition below. Informally, we can think of a
C96 PFA in terms of an algorithm that reads a sequence of input symbols from left to right,
C97 having only finite memory. That is, it can manipulate a finite number of variables with
C98 bounded range, just like an ordinary finite automaton. In addition, a PFA can make coin
C99 flips. As a consequence, the question whether the PFA arrives in an accepting state and
C100 thus accepts a given input word is not a yes/no decision, but it happens with a certain
C101 probability. The language *recognized* (or *represented*) by a PFA is defined by specifying
C102 a probability threshold or *cutpoint* λ . By convention, the language consists of all words
C103 for which the probability of acceptance strictly exceeds λ .

C104 The *PFA Emptiness Problem* is the problem of deciding whether this language is
C105 empty.

C106 This problem is undecidable. There are three different proofs of this theorem in the
C107 literature. The first is by Masakazu Nasu and Namio Honda [23] from 1969. The second
C108 proof, by Volker Claus [8] is loosely related to this proof. A completely independent
C109 proof, which uses a very different approach, was given by Anne Condon and Richard J.
C110 Lipton [10] in 1989, based on ideas of Rūsiņš Freivalds [14] from 1981. The somewhat
C111 intricate history is described in Section 3.

C112 We will present these three proofs in Sections 5, 8, and 4, respectively. The chains
C113 of reductions are shown in Figure 11 in Section 10.2. A self-contained proof of the basic
C114 undecidability result (Proposition 2) takes about 3 pages, see Section 5. The rest of the
C115 paper is devoted to different sharpenings of the undecidability statement, where certain
C116 parameters of the PFA are restricted (Theorems 1–6).

C117 1.1 Formal problem definition

C118 Formally, a PFA is given by a sequence of stochastic *transition matrices* M_σ , one for
C119 each letter σ from the input alphabet Σ . The matrices are $d \times d$ matrices if the PFA has
C120 d states. The start state is chosen according to a given probability distribution $\pi \in \mathbb{R}^d$.
C121 The set of accepting states is characterized by a 0-1-vector $f \in \{0, 1\}^d$.

C122 In terms of these data, the PFA Emptiness Problem with cutpoint λ , whose undecid-
C123 ability we will show, can be formally described as follows.

C124 PFA EMPTINESS. Given a finite set of stochastic matrices $\mathcal{M} \subset \mathbb{Q}^{d \times d}$, a
C125 probability distribution $\pi \in \mathbb{Q}^d$, and a 0-1-vector $f \in \{0, 1\}^d$, is there a
C126 sequence M_1, M_2, \dots, M_m with $M_j \in \mathcal{M}$ such that

$$C127 \quad \pi^T M_1 M_2 \dots M_m f > \lambda ? \quad (1)$$

C128 The most natural choice is $\lambda = \frac{1}{2}$, but the problem is undecidable for any fixed (rational
C129 or irrational) cutpoint λ with $0 < \lambda < 1$. We can also ask $\geq \lambda$ instead of $> \lambda$.

C130 Our results, which we discuss in the next section, show that the PFA Emptiness
C131 Problem remains undecidable under additional restrictions. Table 1 gives an overview
C132 of the various assumptions and constraints on the data. Theorem 1 is essentially due
C133 to Condon and Lipton. Theorem 5 is due to Claus, and Theorem 6 is due to Mika
C134 Hirvensalo [17].

Theorem	π	$ \mathcal{M} $	$M \in \mathcal{M}$	f	acceptance criterion
Thm. 1	$\pi = e_2$	2	<i>input</i>	$f = e_1$	any (Thm. 7)
Thm. 2a	<i>input</i>	52	18×18 , positive	$f \in \{0, 1\}^{18}$	$\geq 1/2$
Thm. 2b	<i>input</i>	53	11×11	$f = e_1$	$> 1/4$
Thm. 3	<i>input</i>	2	572×572	$f = e_1$	$> 1/4$
Thm. 4a	$0 < \pi_q < 1$	52	9×9 , positive	<i>input</i>	$\geq 1/2$
Thm. 4b	$0 \leq \pi_q \leq 1$	52	11×11	<i>input</i>	$> 1/4$
Thm. 5	$\pi = e_2$	5	<i>input</i> , 9×9 , positive	$f = e_1$	$> 1/9$
Thm. 6	$\pi = e_2$	2	<i>input</i> , 20×20 , positive	$f = e_1$	$> 1/20$ or $\geq 1/20$

Table 1: The main characteristics of the data π , \mathcal{M} , and f for different undecidable versions of PFA Emptiness. The data that are not marked as *input* are fixed. The vectors e_1 and e_2 are two standard unit vectors of appropriate dimension.

C135 2 Statement of results

C136 The PFA Emptiness Problem is undecidable even if the start state is a fixed (deterministic)
C137 state, and there is a single accepting state (different from the start state). In this
C138 case, π is a standard unit vector, consisting of a single 1 and otherwise zeros, and likewise,
C139 f is a standard unit vector. The acceptance probability is found in a specific entry (say,
C140 the upper right corner) of the product $M_1 M_2 \dots M_m$.

C141 **Theorem 1.** *For any fixed λ with $0 < \lambda < 1$, the PFA Emptiness Problem (1) with
C142 cutpoint λ is undecidable, even when restricted to instances where \mathcal{M} consists of only
C143 two transition matrices, all of whose entries are from the set $\{0, \frac{1}{2}, 1\}$, and π and f are
C144 standard unit vectors.*

C145 The proof is given in Section 4.

C146 We mention that we don't have to rely on a sharp distinction between $\leq \lambda$ and $> \lambda$,
C147 because the PFA that is constructed in the proof exhibits a strong separation property
C148 (see Theorem 7 in Section 4.3.1, and Section 4.3.3): Either there is a sequence of matrices
C149 for which the product $\pi^T M_1 M_2 \dots M_m f$ exceeds $1 - \varepsilon$, or, for every sequence, the value
C150 is below ε , where ε be chosen arbitrarily close to 0.

C151 The remaining results deal with the case where all matrices in \mathcal{M} are fixed. We
C152 mention that all these results have in the meantime been superseded by stronger results
C153 that use fewer and smaller matrices, see [36, Table 1].

C154 **Definition 1.** *By a binary fraction, we mean a rational number whose denominator is
C155 a power of 2.*

C156 **Theorem 2.**

C157 (a) *There is a fixed set \mathcal{M}' of 52 stochastic matrices of size 18×18 with positive entries*
 C158 *that are multiples of $1/2^{47}$, and a fixed vector $f \in \{0, 1\}^{18}$, for which the following*
 C159 *question is undecidable:*

C160 *Given a probability distribution $\pi \in \mathbb{Q}^{18}$ whose entries are positive binary fractions,*
 C161 *is there a product $M_1 M_2 \dots M_m$, with $M_j \in \mathcal{M}'$ for all $j = 1, \dots, m$, with*

C162
$$\pi^T M_1 M_2 \dots M_m f \geq \frac{1}{2} ?$$

C163 (b) *There is a fixed set \mathcal{M} of 53 stochastic matrices of size 11×11 , all of whose entries*
 C164 *are multiples of $1/2^{47}$, for which the following question is undecidable:*

C165 *Given a probability distribution $\pi \in \mathbb{Q}^{11}$ whose entries are binary fractions, is there*
 C166 *a product $M_1 M_2 \dots M_m$, with $M_j \in \mathcal{M}$ for all $j = 1, \dots, m$, such that*

C167
$$\pi^T M_1 M_2 \dots M_m e_1 > \frac{1}{4} ?$$

C168 *In other words, is the language recognized by the PFA with starting distribution π*
 C169 *and cutpoint $\lambda = \frac{1}{4}$ nonempty?*

C170 In part (b), e_1 denotes the first unit vector in \mathbb{R}^{11} , meaning that there is a single
 C171 accepting state. The proof is given in section 7.5.

C172 Part (b) of the theorem has the acceptance criterion $> \frac{1}{4}$, in line with the conventions
 C173 for a PFA. Part (a) deviates from this convention by using a weak inequality $\geq \frac{1}{2}$, but this
 C174 is rewarded by allowing a stronger assumption: All matrices in \mathcal{M} are strictly positive.

C175 The distinction between the cutpoint values $\frac{1}{2}$ and $\frac{1}{4}$ in parts (a) and (b) is inessential.
 C176 In fact, for all of the Theorems 2–4, the cutpoint can be set to any fixed rational value
 C177 within some range, but then the assumption that all entries are binary fractions must be
 C178 given up, and the size of the matrices must sometimes be increased.

C179 An easier version of Theorem 2b, but with matrices of size 12×12 , is proved in
 C180 Section 6.3 (Proposition 5).

C181 The input alphabet can be reduced to two symbols at the expense of the number of
 C182 states. The proof will be given in section 7.6.

C183 **Theorem 3.** *There is a PFA with 572 states, two input symbols with fixed transition*
 C184 *matrices, all of whose entries are multiples of $1/2^{47}$, and with a single accepting state,*
 C185 *for which the following question is undecidable:*

C186 *Given a probability distribution $\pi \in \mathbb{Q}^{572}$ whose entries are binary fractions, is the*
 C187 *language recognized by the PFA with starting distribution π and cutpoint $\lambda = \frac{1}{4}$ nonempty?*

C188 **More general acceptance.** If each state q is allowed to have an arbitrary probability
 C189 f_q as an “acceptance degree” instead of just 0 or 1, we can also turn things around and
 C190 fix the starting distribution π , but let the values f_q be part of the input. The following
 C191 theorem will be proved in Section 7.3.

C192 **Theorem 4.**

C193 (a) *There is a fixed set \mathcal{M}''' of 52 positive stochastic matrices of size 9×9 and a fixed*
 C194 *starting distribution π , all with positive entries that are multiples of $1/2^{44}$, for which*
 C195 *the following question is undecidable:*

C196 *Given a vector $f \in \mathbb{Q}^9$ whose entries are binary fractions from the interval $[\frac{1}{4}, \frac{5}{8}]$, is*
 C197 *there a product $M_1 M_2 \dots M_m$, with $M_j \in \mathcal{M}'''$ for all $j = 1, \dots, m$, with*

C198
$$\pi^T M_1 M_2 \dots M_m f \geq \frac{1}{2} ?$$

C199 (b) *There is a fixed set \mathcal{M}'' of 52 stochastic matrices of size 11×11 and a fixed starting*
 C200 *distribution π , all of whose entries are multiples of $1/2^{45}$, for which the following*
 C201 *question is undecidable:*

C202 *Given a vector $f \in \mathbb{Q}^{11}$ whose entries are binary fractions from the interval $[0, 1]$, is*
 C203 *there a product $M_1 M_2 \dots M_m$, with $M_j \in \mathcal{M}''$ for all $j = 1, \dots, m$, such that*

C204
$$\pi^T M_1 M_2 \dots M_m f > \frac{1}{4} ?$$

C205 The distinction between parts (a) and (b) is analogous to Theorem 2. This time,
 C206 part (a) has an additional advantage: In addition to the positivity of all data in \mathcal{M} , π ,
 C207 and f , the dimension is reduced from 11 to 9.

C208 **Uniqueness of solutions.** We mention that Theorems 2–4, can be modified such that
 C209 the solutions of the constructed matrix product problem instances are unique if they
 C210 exist, see Section 7.4. In other words, we are guaranteed that the language recognized
 C211 by the PFA contains at most one word. This extension requires a slightly larger number
 C212 of matrices with larger denominators in its entries.

C213 **Smaller transition matrices.** In 1981, Volker Claus [8] investigated what he called
 C214 the (n, k) -bounded Emptiness Problem, for PFAs with at most n states and an alphabet
 C215 of size at most k . He derived the undecidability for certain parameter pairs (n, k) from
 C216 the PCP. In 1981, it was known that the PCP is undecidable with as few as 9 word pairs.
 C217 Meanwhile, we know by results of Neary [24] from 2015 that 5 word pairs are sufficient.
 C218 With this improved bound, the result of Claus reads as follows.

C219 **Theorem 5** (Claus [8, Theorem 6(iii), p. 151]). *The PFA Emptiness Problem (1) is*
 C220 *undecidable for PFAs with a deterministic start state, a single accepting state, and 5*
 C221 *positive transition matrices of size 9×9 , and with cutpoint $\lambda = 1/9$.*

C222 The same approach to find small matrices for which the PFA Emptiness Problem is
 C223 unsolvable was used in 2003 by Blondel and Canterini [3], who concentrated on instances
 C224 with *two* matrices, or in other words, PFAs with a binary input alphabet. These results
 C225 were improved by Hirvensalo [17]¹, who showed in 2007 that the PFA Emptiness Problem
 C226 is undecidable for two transition matrices of size 25×25 . Substituting the improved
 C227 bound of Neary [24] on the number of word pairs for which the PCP is undecidable, five
 C228 states can be saved.

C229 **Theorem 6** ([17]). *The PFA Emptiness Problem (1) is undecidable for PFAs with two*
 C230 *positive transition matrices size 20×20 , a single deterministic start state, a single ac-*
 C231 *cepting state, and with cutpoint $\lambda = 1/20$.*

C232 *The same is true with weak inequality ($\geq 1/20$) as the acceptance criterion.*

C233 We give the proofs of these theorems in Section 8. We mention that the reduction to a
 C234 binary input alphabet for the PFA (i.e., two matrices) was already considered by Claus [8],
 C235 but his results are superseded by Theorem 6. Claus also has results for alphabets of size
 C236 3 and 4 [8, Theorem 6(iii), p. 151]. Conversely, constructions of PFAs with few states
 C237 regardless of the number of matrices are implicit in the proofs of [3] and [17], but they
 C238 are not as strong as Theorem 5.

C239 ¹see also the technical report [16]

3 Preface: history and matrix products

C240 3.1 Three proofs

C241 The study of probabilistic finite automata was initiated by Michael Rabin in 1963 [31].
 C242 While this was an active research area in the 1960's, it is less active today. The first
 C243 proof that PFA Emptiness is undecidable is due to Masakazu Nasu and Namio Honda
 C244 from 1969 [23, Theorem 21, p. 270]. It proceeds through a series of lemmas with tricky
 C245 constructions, showing that more and more classes of languages, including certain types
 C246 of context-free languages, can be recognized by a PFA. Eventually, the undecidability of
 C247 the PFA Emptiness Problem is derived from Post's Correspondence Problem (PCP, see
 C248 Section 5.2). The proof is reproduced in the final part of a monograph by Azaria Paz
 C249 from 1971 [30, Theorem 6.17 in Section IIIB, p. 190]. The presentation is quite close
 C250 to the original, but very much condensed (and it never cares to mention the PCP by
 C251 name!). I suppose, as the result was still recent when the book was written, it was the
 C252 culmination point of the monograph. It appears as part of the last theorem of the book,
 C253 before a brief final chapter on applications and generalizations. In the literature, as far
 C254 as I have surveyed it, the result is almost universally misattributed to Paz, although Paz
 C255 gave credit to Nasu and Honda (not very specifically, however) in the closing remarks
 C256 of the chapter [30, Section IIIB.7, Bibliographical notes, p. 193].² A simplified version
 C257 of this proof appears in a textbook of Volker Claus from 1971 [7, Satz 28, p. 157] in
 C258 German, with proper attribution to Nasu and Honda.

C259 A second proof is due to Claus [8] from 1981. It has been practically forgotten
 C260 until now. It also takes the PCP as the starting point, but it proceeds via products of
 C261 integer matrices, which were first used in this context by Mike Paterson [29] in 1970. As
 C262 mentioned earlier, Claus was interested in constraints on the number of states and the size
 C263 of the input alphabet for which the Emptiness Problem for PFAs remains undecidable.
 C264 These results were later rediscovered and improved by Blondel and Canterini [3] in 2003
 C265 and further improved by Hirvensalo in 2007 [17].

C266 A completely independent proof was sketched by Anne Condon and Richard Lipton
 C267 in 1989 [10]. It arose as an auxiliary result for their investigation of space-bounded
 C268 interactive proofs. Condon and Lipton based their reduction on the undecidability of the
 C269 Halting Problem for 2-Counter Machines (2CM), see Section 4 below.

C270 3.2 Interlude: Other problems on matrix products

C271 As the formulation (1) shows, the PFA Emptiness Problem is about products of matrices
 C272 that can be taken from a given set \mathcal{M} . There are other problems of this type, whose
 C273 undecidability comes down to PFA Emptiness: For example, the *joint spectral radius* of
 C274 a set \mathcal{M} of $d \times d$ matrices is

$$C275 \limsup_{m \rightarrow \infty} \max_{A_1, A_2, \dots, A_m \in \mathcal{M}} \sqrt[m]{\|A_1 A_2 \dots A_m\|},$$

C276 where $\|\cdot\|$ denotes an arbitrary norm. In 2000, Blondel and Tsitsiklis [4] proved, based
 C277 on the PFA Emptiness Problem, that it is undecidable whether the joint spectral radius
 C278 of a finite set of rational matrices exceeds 1.

C279 This has recently been generalized in the analysis of the growth rate of *bilinear sys-*
 C280 *tems*, see Matthieu Rosenfeld [32] and Vuong Bui [5, 6]. The study of bilinear systems was
 C281 initiated for a special case of such a system in Rote [34] in the context of a combinatorial
 C282 counting problem. Corresponding decidability questions are discussed in Rosenfeld [33]

C283 ²“The rest of that section beginning from Exercise 6.9 and on is based on the work of Nasu and Honda (1970).” The year 1970 is a mistake.

C284 and Bui [6, Chapter 6]. These connections were my motivation for starting the investi-
 C285 gations about the PFA Emptiness Problem.

C286 In fact, Theorem 4a, which strengthens the undecidability result of PFA Emptiness
 C287 to *positive* transition matrices, can be used to resolve a conjecture of Bui [6, Conjecture
 C288 6.7], by adapting the reduction of Blondel and Tsitsiklis [4]: Already for two *positive*
 C289 matrices, it is undecidable to check if their joint spectral radius is larger than 1.

C290 3.3 ... back to the proofs of PFA Emptiness:

C291 In 2000, Blondel and Tsitsiklis [4], not being aware of the paper of Claus [8] from 1981,
 C292 could arguably complain that *a complete proof that PFA Emptiness is undecidable cannot*
 C293 *be found in its entirety in the published literature*. Since then, Condon and Lipton’s proof
 C294 has been published in sufficient detail in other papers, for example by Madani, Hanks,
 C295 and Condon [19, Sec. 3.1 and Appendix A] in 2003. Moreover, in the publication list
 C296 on Anne Condon’s homepage, the entry for the Condon–Lipton conference paper [10]
 C297 from 1989 links to a 22-page manuscript, dated November 29, 2005.³ According to the
 C298 metadata, the file was generated on that date by the dvips program from a file called
 C299 “journalsub.dvi”. This manuscript also gives the proof in detail. Condon and Lipton’s
 C300 proof, which is based on ideas of Freivalds, is conceptually simple and illuminating. The
 C301 current article originated from lecture notes about this proof.

C302 Meanwhile, I struggled with Nasu and Honda’s article and tried to penetrate through
 C303 its rendition in Paz [30], which proceeds through a cascade of definitions and lemmas
 C304 that stretch over the whole book. When I had already acquired a rough understanding
 C305 of some crucial ideas, I was lucky to hit upon the undecidability proof in the textbook
 C306 of Claus [7, Satz 28, p. 154–157], which is considerably simplified. The result in [7]
 C307 is weaker, because the number of input symbols is the number k of word pairs of the
 C308 PCP, whereas Nasu and Honda establish undecidability already for an input alphabet of
 C309 size 2. It is, however, easy to reduce the input alphabet, see Lemma 3. (Nasu and Honda’s
 C310 technique for achieving this reduction is considerably more involved, see Section 11.4 and
 C311 Appendix A.)

C312 After finishing a previous version of this note, I learned about another undecidability
 C313 proof of the PFA Emptiness Problem by Gimbert and Oualhadj from 2010 [15, Sec-
 C314 tion 2.1]. Compared to the Nasu–Honda proof, it treats the equality test $\phi(a) = \psi(a)$
 C315 (Section 5.3) differently, but otherwise it is similar to the proof that I present (Proposi-
 C316 tion 2 in Section 5). The authors misattribute the main auxiliary result of their proof to
 C317 a paper of Alberto Bertoni from 1975 [1]. More details are given in Section 5.5.1.

C318 By tracing the literature back from [15], I discovered the paper of Claus [8] from 1981,
 C319 which is about the Emptiness Problem expressly for PFAs with few states and with small
 C320 input alphabet. It takes the opposite route: Rather than using PFAs to prove results
 C321 about matrix products, it starts from problems involving products of integer matrices
 C322 and converts them to PFAs

C323 3.4 Overview

C324 In this note, I try to present the best parts of the three proofs in a self-contained way.
 C325 I use slightly different terminology, and some details vary from constructions found else-
 C326 where. I have preferred concrete formulations with particular values of the parameters,
 C327 illustrating them with examples. Generalizations to arbitrary parameters are treated as
 C328 an afterthought. I have made an effort to streamline the proofs. In particular, the com-
 C329 plete Nasu–Honda–Claus proof leading to the main undecidability result of Proposition 2

C330 ³<https://www.cs.ubc.ca/~condon/papers/condon-lipton89.pdf>, accessed 2024-05-01.

C331 takes only 3 pages (Section 5, pp. 16–19), and I encourage the reader to jump directly
 C332 to this section. In later parts, I will incrementally introduce new ideas that decrease the
 C333 number of states or deal with variants of the problem, and the treatment becomes more
 C334 technical. For reference, I review the original Nasu–Honda proof in Appendix A.

C335 3.5 Comparison of the proofs

C336 The proofs use different ideas, and they have different merits: Condon and Lipton’s proof
 C337 leads to an arbitrarily large gap between accepting and rejecting probabilities (Theorem 7
 C338 and Section 4.3.3)⁴ and it is easy to restrict the input alphabet to 2 symbols (Theorem 1).
 C339 The number of states is beyond control. While the constructions in Condon and Lipton’s
 C340 proof use a high-level description of a PFA as a randomized algorithm, the proof of Nasu
 C341 and Honda encourages to work with the transition matrices directly, and consequently,
 C342 allows a finer control over the number of states.

C343 Moreover, by looking at the reductions in detail, one can even show undecidability
 C344 of the Emptiness Problem for a *fixed* PFA with 11 states and an input alphabet of size
 C345 53, where the only variable input is the starting distribution (Theorem 2b). This and
 C346 similar sharpenings of the undecidability statement are the contributions of this paper
 C347 in terms of new results, and we hope they might find other applications. A variation of
 C348 the problem allows as few as 9 states (Theorem 4a).

C349 An alternative approach is somewhat similar in spirit, but it takes a detour via integer
 C350 matrices, which are only in the end converted to stochastic matrices via Turakainen’s
 C351 Theorem [40] (see the introduction of Section 7). This also allows as few as 9 states
 C352 (Theorem 5).

C353 The distinction between the two main proof approaches is highlighted for a particular
 C354 example, the language $\{a^i b^i \# \mid i \geq 0\}$, in Section 10.1.

C355 4 The Condon–Lipton proof via 2-counter machines

C356 This section presents the proof of Condon and Lipton [10] from 1989, leading to Theo-
 C357 rem 1.

C358 A *counter machine* has a finite control, represented by a state q from a finite set Q ,
 C359 and a number of nonnegative counters. There is a designated start state and a designated
 C360 halting state. Such a machine operates as follows. At each step, it checks which counters
 C361 are zero. Depending on the outcome of these tests and the current state q , it may
 C362 increment or decrement each counter by 1, and it enters a new state.

C363 A counter machine with as few as two counters (a 2CM) is as powerful as a Turing
 C364 machine. This was first proved by Marvin Minsky [21] in 1961 and is by now textbook
 C365 knowledge [18, Theorem 7.9].⁵ The question whether such a 2-counter machine halts if
 C366 it is started with both counter values at 0 is undecidable.

C367 ⁴However, there is a general technique by which the acceptance by strict inequality $> \lambda$ can be
 C368 amplified to a gap between $\leq 1/2$ and arbitrarily close to 1, see Section 11.3 and the remark at the end
 C369 of Section 10.1.

C370 ⁵The usual way to simulate a Turing machine by a 2CM proceeds in three easy steps: (i) A two-sided
 C371 infinite tape can be simulated by two push-down stacks. (ii) A push-down stack can be simulated by
 C372 two counters, interpreting the stack contents as digits for an appropriate radix that is large enough
 C373 to accommodate the stack alphabet; two counters are necessary to perform multiplication and division
 C374 by the radix. (iii) Any number of counters can be simulated by two counters, representing the values
 C375 a, b, c, d, \dots of the counters as a product $2^a 3^b 5^c 7^d \dots$ of prime powers. See [https://en.wikipedia.org](https://en.wikipedia.org/wiki/Counter_machine#Two-counter_machines_are_Turing_equivalent_(with_a_caveat))
 C376 [/wiki/Counter_machine#Two-counter_machines_are_Turing_equivalent_\(with_a_caveat\)](https://en.wikipedia.org/wiki/Counter_machine#Two-counter_machines_are_Turing_equivalent_(with_a_caveat)), accessed
 C377 2024-04-13.

C378 Denoting by q_i, l_i, r_i the state and the values of the two counters after i steps, an
 C379 accepting computation with m steps can be written as follows:

C380
$$l_0, r_0, q_0, l_1, r_1, q_2, l_2, r_2, q_3, \dots, l_{m-1}, r_{m-1}, q_m$$

C381 To turn it into an input for a finite automaton, we encode it as a word A over the alphabet
 C382 $Q \cup \{0, 1, \#\}$ with an end marker $\#$:

C383
$$A = 0^{l_0} 1^{r_0} q_0 0^{l_1} 1^{r_1} q_1 0^{l_2} 1^{r_2} q_2 \dots 0^{l_m} 1^{r_m} q_m \# \tag{2}$$

C384 There are some conditions for an accepting computation that a deterministic finite au-
 C385 tomaton can easily check: Does the word conform to this format? Do the state transitions
 C386 follow the rules? Is $l_0 = r_0 = 0$? Is the initial and the final (halting) state correct? We
 C387 refer to these checks as the *formal checks*.

C388 The only thing that a finite automaton cannot check is the consistency of the counters,
 C389 for example, whether l_{i+1} is equal to l_i , or $l_i + 1$, or $l_i - 1$, as appropriate.

C390 For this task, we use the probabilistic capacities of the PFA. If there is an accepting
 C391 computation A of the form (2) for the counter machine, we feed this computation as
 C392 input to the PFA again and again. In other words, we input the word A^t for a large
 C393 enough t . We will set up the PFA in such a way that there is a strong separation of
 C394 probabilities: It will accept this input with probability at least 0.99. On the other hand,
 C395 if there is no accepting computation, then every input will be rejected with probability
 C396 at least 0.99.

C397 **4.1 The Equality Checker**

C398 As an auxiliary procedure, we study a PFA that reads words of the form $a^i b^j \#$. The goal
 C399 is to “decide” whether $i = j$. We call this procedure the *Equality Checker*. There are
 C400 three possible outcomes, “Different”, “Same”, or “Undecided”.

C401 The PFA simulates a competition between two players D and S (“Different” and
 C402 “Same”, or “Double” and “Sum”), as shown in Figure 1. There are four unbiased coins of
 C403 different colors.

- C404
 - Player D flips the red coin twice for each a and the orange coin twice for each b .
- C405
 - Player S flips the blue coin and the green coin for each input symbol (a or b).

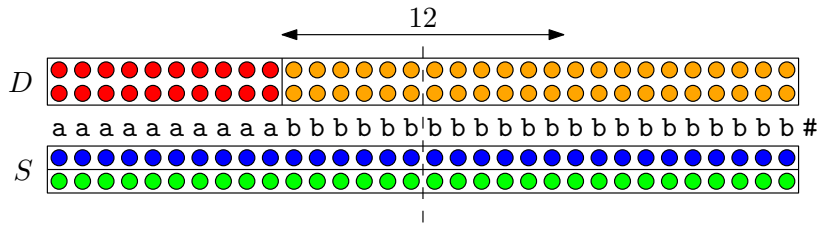


Figure 1: The coin flips for the input $a^{10}b^{22}\#$

C406 In addition, the PFA keeps track of the difference $i - j$ modulo 12. If $i \not\equiv j \pmod{12}$,
 C407 the PFA declares the outcome to be “Different”.

C408 If $i \equiv j \pmod{12}$, the outcome of the game is defined as follows. We call a coin *lucky*
 C409 if it always came up heads.

- C410
 - If D has a lucky coin and S has no lucky coin, declare “Different”.
- C411
 - If S has a lucky coin and D has no lucky coin, declare “Same”.

C412 • Otherwise, declare “Undecided”.

C413 Since i and j are usually large, lucky means *extremely lucky*. Thus, the first two events
 C414 are very rare, and the outcome will almost always be “Undecided”. The outcome of the
 C415 Equality Checker is illustrated in Figure 2 and described in the following lemma.

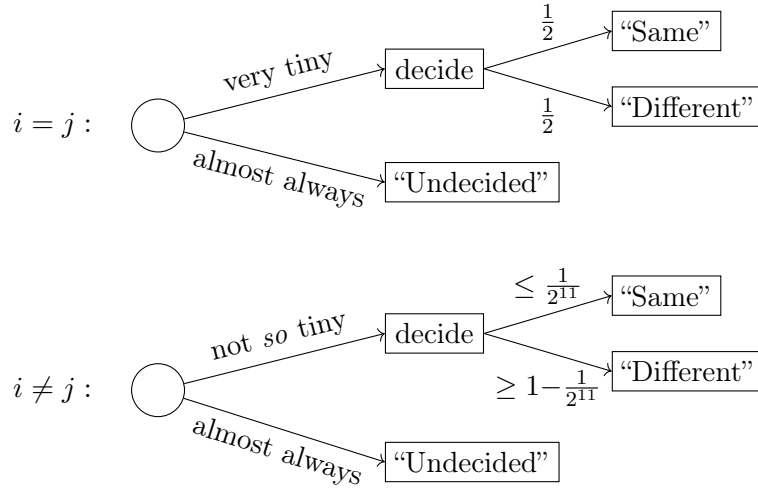


Figure 2: The behavior of the Equality Checker, assuming $i \equiv j \pmod{12}$

C416 **Lemma 1.**

- C417 • If $i = j$, $\Pr[\text{“Different”}] = \Pr[\text{“Same”}]$.
- C418 • If $i \neq j$, $\Pr[\text{“Different”}] \geq 2^{11} \cdot \Pr[\text{“Same”}]$.

C419 *Proof.* The first statement is clear, since each coin is flipped $2i$ times, and the situation
 C420 between D and S is symmetric.

C421 Assume that $i \neq j$. If $i \not\equiv j \pmod{12}$, then $\Pr[\text{“Same”}] = 0$, and we are done.

C422 Otherwise, $|i - j| \geq 12$, and the smaller of i and j , say i , is at most $i \leq \frac{i+j}{2} - 6$. Then
 C423 the red coin is flipped at most $2i \leq i + j - 12$ times. Thus,

C424
$$\Pr[D \text{ has a lucky coin}] \geq \Pr[\text{the red coin was lucky}] \geq 1/2^{i+j-12} \quad (3)$$

C425 The blue and the green coin was each flipped $i + j$ times, and hence

C426
$$\Pr[S \text{ has a lucky coin}] \leq$$

 C427
$$\Pr[\text{the blue coin was lucky}] + \Pr[\text{the green coin was lucky}] \leq 2/2^{i+j} \quad (4)$$

C428 The ratio $\Pr[D \text{ lucky}]/\Pr[S \text{ lucky}]$ between (3) and (4) is at least 2^{11} . From each of
 C429 these probabilities, we have to subtract the (small) probability that both S and D have
 C430 a lucky coin, but this tilts the ratio between “Different” and “Same” even more in D ’s
 C431 favor. Formally:

C432
$$\frac{\Pr[\text{“Different”}]}{\Pr[\text{“Same”}]} = \frac{\Pr[D \text{ lucky}] - \Pr[D \text{ lucky and } S \text{ lucky}]}{\Pr[S \text{ lucky}] - \Pr[D \text{ lucky and } S \text{ lucky}]} > \frac{\Pr[D \text{ lucky}]}{\Pr[S \text{ lucky}]} \geq 2^{11} \quad \square$$

C433 Since the algorithm only needs to count up to 11 and to maintain a few flags, it is
 C434 clear that it can be carried out by a PFA.⁶

C435 ⁶As an exercise, the reader may try to work out the required number of states. The outcomes should

C436 **4.2 Correctness Test: checking a 2CM computation**

C437 Recall that we wish to check a description of a computation of the form

C438
$$A = 0^{l_0} 1^{r_0} q_0 0^{l_1} 1^{r_1} q_1 0^{l_2} 1^{r_2} q_2 \dots 0^{l_n} 1^{r_n} q_n \# .$$

C439 The Equality Checker can be adapted to look at, say, two consecutive zero blocks 0^{l_i}
 C440 and $0^{l_{i+1}}$ of a computation that represent the values of the counter l and check whether
 C441 $l_{i+1} = l_i$. It can also be adapted to check $l_{i+1} = l_i + 1$, or $l_{i+1} = l_i - 1$, as appropriate
 C442 for the state q_i and the results of the zero test of l_i and r_i . The guarantees of Lemma 1
 C443 about the outcome remain valid.

C444 We run independent Equality Checkers for each relation between two consecutive
 C445 values l_i and l_{i+1} , as well as r_i and r_{i+1} , of a computation A . In total, these are $2n$
 C446 Equality Checkers. In the schematic drawing of Figure 3, the outcomes of the Equality
 C447 Checkers are shown as a row of boxes. Typically, most of them will be “Undecided”,
 C448 with a few interspersed “Same” and “Different” results (proportionally much fewer than
 C449 shown in the first example row). We are interested in the rare cases when all outcomes
 C450 are “Same”, or all “Different”.

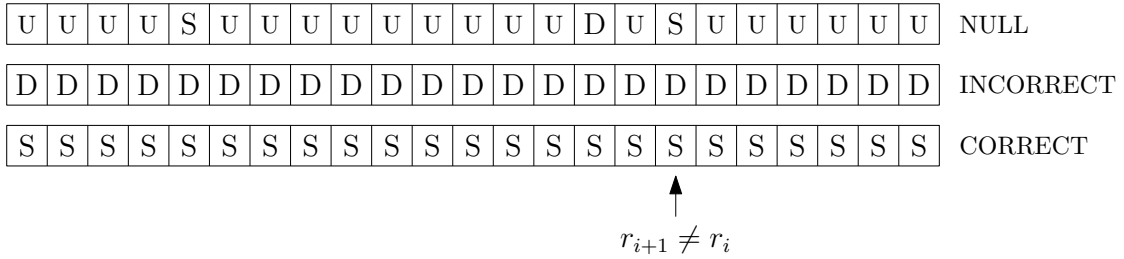


Figure 3: The Correctness Test for a computation A , and a hypothetical position where equality does not hold.

C451 The output of these Equality Checkers is aggregated into a *Correctness Test* as fol-
 C452 lows: We report the output “CORRECT” if *all* Equality Checkers report “Same”, and we
 C453 report the output “INCORRECT” if *all* Equality Checkers report “Different”. Otherwise,
 C454 we report “NULL”.

C455 To compute this result, only four independent Equality Checkers have to run simulta-
 C456 neously: While reading the input, the current block lengths l_i and r_i have to be compared
 C457 with the preceding and the next values. Thus, the computation can be implemented by
 C458 a PFA, with finitely many states. (Looking more carefully, one sees that actually, only
 C459 three Equality Checkers are active at the same time: For example, when reading 1^{r_i} , the
 C460 Equality Check between $0^{l_{i-1}}$ and 0^{l_i} has already been completed.)

C461 **Lemma 2.** *Suppose that a computation A of the form (2) passes all formal checks.*

C462 *If A represents an accepting computation,*

C463
$$\Pr[\text{“INCORRECT”}] = \Pr[\text{“CORRECT”}].$$

C464 *If A does not represent an accepting computation,*

C465
$$\Pr[\text{“INCORRECT”}] \geq 2^{11} \cdot \Pr[\text{“CORRECT”}].$$

C466 be represented by a partition of the states into four classes, including a category “Rejected” for inputs
 C467 that don’t adhere to the format $a^i b^j \#$. A literal and naive implementation that simply keeps track of
 C468 every lucky and unlucky coin and sets a flag when a b is seen (this is the only thing that needs to
 C469 be remembered in order to check the syntax, except for a final state change on reading $\#$) would need
 C470 $2^5 \times 12 + 4 = 388$ states. By excluding impossible combinations of flags and with some other tricks like
 C471 merging states whose distinction is irrelevant (see Section 5.6), I managed to do it with 173 states. If
 C472 the PFA can trust that the input has the correct format, 108 states suffice.

C473 *Proof.* The probability for “CORRECT” is the product of the probabilities that each
 C474 Equality Checker results in “Same”, and analogously, for “INCORRECT” and “Different”.

C475 If A represents an accepting computation, then all Equality Checkers are balanced
 C476 between “Same” and “Different”, and the result is clear. Otherwise, there is at least one
 C477 position (marked by an arrow in Figure 3) where an error occurs, and the probability
 C478 for “Different” is at least 2^{11} times larger than for “Same”, according to Lemma 1. In all
 C479 other Equality Checkers, the probability is either balanced or it gives a further advantage
 C480 for “Different”. Thus, the product of the probabilities is at least 2^{11} times larger for “all
 C481 Different” than for “all Same”. \square

C482 4.3 Third-level aggregation: processing the whole input

C483 An Equality Checker aggregates the results of many coin flips into an output “Same”,
 C484 “Different”, or “Undecided”. We have further aggregated the result of many Equality
 C485 Checkers into a Correctness Test for the word A (with output “CORRECT”, “INCOR-
 C486 RECT”, or “NULL”). We add yet another level of aggregation in order to decide whether
 C487 the PFA should accept the input word. As mentioned, we feed the PFA with a huge
 C488 number of copies of an accepting computation A . Each copy of A is subjected to the
 C489 Correctness Test.

C490 If we take the first definite result (“CORRECT” or “INCORRECT”) as an indication
 C491 whether to accept or reject the input, we get an acceptance probability close to $1/2$ on
 C492 a valid input. (It is a little less than $1/2$ because of the chance that the input runs out
 C493 before a definite answer is obtained.) On the other hand, if no accepting computation
 C494 exists, any input must consist of “fake” computations. The algorithm will recognize this
 C495 and reject with probability at least $1 - 1/2^{11}$.

C496 4.3.1 Increasing the acceptance probability

C497 We modify the rules to make the acceptance probability larger, at the expense of the
 C498 rejection probability for fake inputs. We determine the overall result as follows. As
 C499 soon as a Correctness Test yields “CORRECT”, we accept the input. However, in order
 C500 to reject the input, we wait until we have received 10 answers “INCORRECT” before
 C501 receiving an answer “CORRECT”. If the end of the input is reached before any of these
 C502 events happens, this also leads to rejection. Of course, we also reject the input right
 C503 away if any of the formal checks fails.

C504 **Theorem 7.** *If there is an accepting computation A for the 2CM, then the PFA accepts*
 C505 *the input A^t , for sufficiently large t , with probability more than 0.99.*

C506 *If there is no accepting computation, then the PFA rejects every input with probability*
 C507 *at least 0.99.*

C508 *Proof.* If A is an accepting computation, the distribution between “CORRECT” and “IN-
 C509 CORRECT” is fair. Thus, the probability of receiving 10 outputs “INCORRECT” before
 C510 receiving an output “CORRECT” is $1/2^{10} < 0.001$. To this we must add the probability
 C511 of rejection because the input runs out before receiving an output “CORRECT”, but this
 C512 can be made arbitrarily small by increasing t .

C513 If there is no accepting computation, then “INCORRECT” has an advantage over
 C514 “CORRECT” by a factor at least 2^{11} . If the input runs out before a decision is reached,
 C515 this is in the favor of rejection. Otherwise, the probability of receiving 10 outputs “IN-
 C516 CORRECT” before receiving an output “CORRECT” is at least

$$\begin{aligned}
 \left(\frac{2^{11}}{2^{11} + 1}\right)^{10} &= \left(1 - \frac{1}{2^{11} + 1}\right)^{10} \geq \left(1 - \frac{1}{2000}\right)^{10} \approx 1 - \frac{1}{200} = 0.995. \quad \square \\
 \text{C517}
 \end{aligned}$$

C518 If the 2CM halts, there is an accepting computation A . (A is unique since the 2CM
C519 is deterministic.) In this situation, the language recognized by the PFA with cutpoint
C520 $\lambda = \frac{1}{2}$ contains the set $\{A^t \mid t \geq t_0\}$ for some large t_0 . Otherwise, the language is empty.

C521 As a consequence, checking whether the language accepted by a PFA is empty is
C522 undecidable.

C523 4.3.2 Who is afraid of small probabilities?

C524 As an exercise, we estimate the necessary number t of repetitions of A . Suppose that
C525 the accepting computation A has m transitions. Then the counter values l_i and r_i are
C526 also bounded by m . The probability of the outcome “Same” in the Equality Checker is
C527 roughly 2^{-m} , and the probability that all $2m$ Equality Checkers for the computation A
C528 yield “Same”, leading to the answer “CORRECT”, is roughly $(2^{-m})^{2m} = 4^{-m^2}$.

C529 We want the probability that none of t experiments gets the answer “CORRECT” to
C530 be ≤ 0.009 (the difference between the bound $0.001 > 1/2^{10}$ established in the proof of
C531 Theorem 7 and the target tolerance 0.01):

$$C532 \quad (1 - 4^{-m^2})^t \leq 0.009$$

C533 Since $1 - 4^{-m^2} \approx \exp(-4^{-m^2})$, we need t to be roughly $5 \cdot 4^{m^2}$.

C534 This dependence on the runtime m of the 2-counter machine does not appear so
C535 terrible; however, when considering the overhead of simulating a Turing machine (see
C536 footnote 5), the dependence blows up to a triply-exponential growth in terms of the
C537 runtime of a Turing machine.

C538 4.3.3 Boosting the decision probabilities

C539 We can boost the decision probabilities beyond 0.99 to become arbitrarily close to 1. We
C540 simply run an odd number of copies of the PFA simultaneously and take a majority vote.

C541 Alternatively, we can adjust the parameters. The number K of times that we wait
C542 for “INCORRECT” before rejecting the input can be increased above $K = 10$. As a
C543 compensation, we have to increase the modulus G (we have chosen $G = 12$) by which
C544 i and j are compared in the Equality Checker. The acceptance probability in case of a
C545 valid input increases to become arbitrarily close to $1 - 1/2^K$, and the rejection probability
C546 for an invalid input is at least $(1 - 1/2^{G-1})^K$.

C547 In summary, for any $\varepsilon > 0$ we can construct the PFA in such a way that it either
C548 accepts *some word* with probability at least $1 - \varepsilon$, or there is *no word* that it accepts
C549 with probability larger than ε . This does not mean that there cannot be words whose
C550 acceptance probability is between those ranges, for example close to $1/2$. Candidates for
C551 such words are the words A^t where t is slightly too small.⁷

C552 4.4 Summing up the proof of Theorem 1

C553 We have described the algorithm for the PFA verbally as a probabilistic algorithm, keep-
C554 ing in mind the finiteness constraints of a finite automaton. Eventually, this algorithm
C555 must be translated into a set of states and transition matrices. Theorem 1 puts some
C556 extra constraints on the PFAs whose emptiness is undecidable.

C557 ⁷In fact, it is impossible to avoid the neighborhood of $1/2$ except for very simple languages: Rabin [31,
C558 Theorem 3] showed in 1963 that an open gap interval (p_1, p_2) of positive length, such that the acceptance
C559 probability never falls in this gap, can only exist if, for a cutpoint λ in this interval, the recognized
C560 language is regular, see also [30, Theorem 2.3 in Section IIIB, p. 160] or [7, §3.2.2, pp. 112–115]. Such a
C561 cutpoint λ is called an *isolated cutpoint*.

C562 **Theorem 1.** For any fixed λ with $0 < \lambda < 1$, the PFA Emptiness Problem (1) with
 C563 cutpoint λ is undecidable, even when restricted to instances where \mathcal{M} consists of only
 C564 two transition matrices, all of whose entries are from the set $\{0, \frac{1}{2}, 1\}$, and π and f are
 C565 standard unit vectors.

C566 *Proof.* The extra constraints can be easily fulfilled:

C567 (a) We encode the input A with a fixed-length binary code for the original input
 C568 alphabet $Q \cup \{0, 1, \#\}$. This means that the set \mathcal{M} can be restricted to only two matrices.
 C569 (Lemma 3 in Section 7.6 below treats this transformation more formally.)

C570 (b) By padding the input, we can ensure that the PFA algorithm needs to toss at
 C571 most one coin per input symbol, and thus the entries of the matrices can be restricted
 C572 to $0, \frac{1}{2}, 1$.⁸ In the algorithm as described, only 16 coin tosses are necessary per input
 C573 character (four coins per Equality Checker running at any point in time). Thus we
 C574 simply pad each codeword in the binary code with 15 zeros.

C575 (c) Our algorithm does not need to make any coin flips before reading the first symbol.
 C576 Thus, we can fix the start state to be a deterministic state.

C577 (d) Finally, a single accepting state is enough: As soon as the algorithm has decided
 C578 to accept the input, it will stay committed to this decision. The accepting state is an
 C579 absorbing state, and there is another absorbing state for rejection. In terms of vectors,
 C580 both the starting distribution π and the characteristic vector f of accepting states are
 C581 standard unit vectors. (Since the empty input is not accepted, the accepting state is
 C582 distinct from the start state, and we can arrange the states so that the acceptance
 C583 probability is found in the upper right corner of the product $M_1 M_2 \dots M_m$.) \square

C584 The dimension of the matrices M_i will be huge. It is to the largest extent determined
 C585 by the number of states of the 2CM, and this is not under control.

C586 4.5 History of ideas

C587 Condon and Lipton credit the main ideas of their proof to Rūsiņš Freivalds [14], who
 C588 studied the Emptiness Problem for probabilistic 2-way finite automata in 1981 (unaware
 C589 of Nasu and Honda’s earlier work). In particular, Freivalds developed the idea of a
 C590 competition between two players to recognize the language $\{a^i b^i \mid i \geq 0\}$ (Section 4.1),
 C591 and aggregating the results of these competitions into “macrocompetitions” (Section 4.2).
 C592 A 2-way automaton can move the input head back and forth over the input, and thus
 C593 process the input as often as it wants. Freivalds claimed that the Emptiness Problem for
 C594 such automata is undecidable [14, Theorem 4]; he gives only a hint that the reduction
 C595 should be from the PCP (Post’s Correspondence Problem, see Section 5.2), without any
 C596 details how to connect “macrocompetitions” with the PCP. I have not been able to come
 C597 up with an idea how the proof would proceed.

C598 For our present case of a (1-way) finite automaton, the repeated scan of the input is
 C599 not possible; it is replaced by providing an input which consists of many repetitions of
 C600 the same word.

C601 5 The Nasu–Honda–Claus proof via Post’s Correspondence Problem

C602 This section presents the proof of Nasu and Honda [23] from 1969 in the version of Claus
 C603 [7] from 1971, leading to the undecidability results in Propositions 1–4, which are then
 C604 strengthened to Theorems 2–4 in the rest of the paper.

C605 ⁸PFA’s with this restricted set of probabilities are called *simple PFA’s* in [15, Definition 2].

C606 **5.1 The binary PFA**

C607 For a string $u \in \{0, 1\}^*$, we denote by $(u)_2$ the numeric value of u when it is interpreted
 C608 as a binary number, and we write $|u|$ for the length of u . We define the stochastic matrix

C609
$$B(u) := \begin{pmatrix} 1 - \frac{(u)_2}{2^{|u|}} & \frac{(u)_2}{2^{|u|}} \\ 1 - \frac{(u)_2+1}{2^{|u|}} & \frac{(u)_2+1}{2^{|u|}} \end{pmatrix}, \text{ for example } B(00110) = \begin{pmatrix} \frac{26}{32} & \frac{6}{32} \\ \frac{25}{32} & \frac{7}{32} \end{pmatrix}.$$

C610 These matrices fulfill the remarkable multiplication law

C611
$$B(u)B(u') = B(u'u), \tag{5}$$

C612 which can be confirmed by a straightforward calculation. Note the reversed order of the
 C613 factors.

C614 Note that the top right entry $\frac{(u)_2}{2^{|u|}}$ of the matrix $B(u)$ is the value $0.u$ when interpreted
 C615 as a binary fraction; in our example, where $u = 00110$, we have $0.u = (0.00110)_2 = \frac{6}{32} =$
 C616 $(0.0011)_2$. We will continue to use the convenient notation $0.u$ for this. As we see, trailing
 C617 zeros don't influence the value of $0.u$, and we have to careful about this.

C618 **5.2 Post's Correspondence Problem (PCP)**

C619 In the *Post Correspondence Problem* (PCP), we are given a list of pairs of words $(v_1, w_1),$
 C620 $(v_2, w_2), \dots, (v_k, w_k)$. The problem is to decide if there is a nonempty sequence $a_1 a_2 \dots a_m$
 C621 of indices $a_i \in \{1, 2, \dots, k\}$ such that

C622
$$v_{a_1} v_{a_2} \dots v_{a_m} = w_{a_1} w_{a_2} \dots w_{a_m}$$

C623 This is one of the well-known undecidable problems.⁹ It is no restriction to fix the
 C624 alphabet to $\{0, 1\}$, since every alphabet can be encoded in binary.

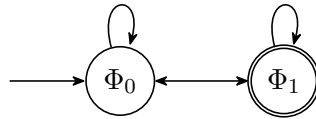


Figure 4: The binary automaton with acceptance probability ϕ

C625 Let us look at the first sequence of words v_1, \dots, v_k . We construct a PFA with input
 C626 alphabet $\{1, 2, \dots, k\}$ and two states Φ_0 and Φ_1 , see Figure 4. The transition matrices
 C627 are $M_i = B(v_i)$. We take Φ_0 as the start state and Φ_1 as the accepting state. Then the
 C628 acceptance probability of the word $a = a_1 a_2 \dots a_m$ is found in the upper right corner
 C629 of the product $M_{a_1} M_{a_2} \dots M_{a_{m-1}} M_{a_m}$ of the corresponding transition matrices, and it
 C630 follows from (5) that this is

C631
$$\phi(a) = 0.v_{a_m} v_{a_{m-1}} \dots v_{a_2} v_{a_1}. \tag{6}$$

C632 We can build an analogous PFA for the other sequence of words w_1, \dots, w_k , and then
 C633 the acceptance probability of a will be

C634
$$\psi(a) = 0.w_{a_m} w_{a_{m-1}} \dots w_{a_2} w_{a_1}. \tag{7}$$

C635 Due to the swapping of the factors in the multiplication law (5), the words are con-
 C636 catenated in (6) and (7) in reverse order, but this cosmetic change does not affect the

⁹A reduction from the Halting Problem for Turing Machines to a closely related problem, the *Modified*
 C637 Post Correspondence Problem (see Section 5.7) is described in detail in Sections 6.1–6.2.

C638 undecidability of the PCP. Thus the PCP comes down to the question whether there is
 C639 a nonempty word a with equal acceptance probabilities $\phi(a) = \psi(a)$ in the two PFAs.

C640 We have to be careful because of the *trailing zeros issue*: Trailing zeros don't change
 C641 the probabilities (6) and (7). An easy way to circumvent this problem is to add a 1 after
 C642 every symbol of every word, thus doubling the length of the words. This ensures that
 C643 there are no trailing zeros that could go unnoticed.

C644 5.3 Testing equality of probabilities

C645 For recognizing the words a with $\phi(a) = \psi(a)$, there is a construction of a PFA that does
 C646 the job. It is based on the identity

$$C647 \quad \frac{1}{2}\phi\psi + \frac{1}{4}(1 - \phi^2) + \frac{1}{4}(1 - \psi^2) = \frac{1}{2} - \frac{1}{4}(\phi - \psi)^2. \quad (8)$$

C648 We will build a PFA for each term $\phi\psi$, $1 - \phi^2$, $1 - \psi^2$ on the left, and we will mix them in
 C649 the right proportion. As the right-hand side shows, we have then (almost) achieved our
 C650 goal: The acceptance probability achieves its maximum value $\frac{1}{2}$ only for $\phi(a) = \psi(a)$.¹⁰

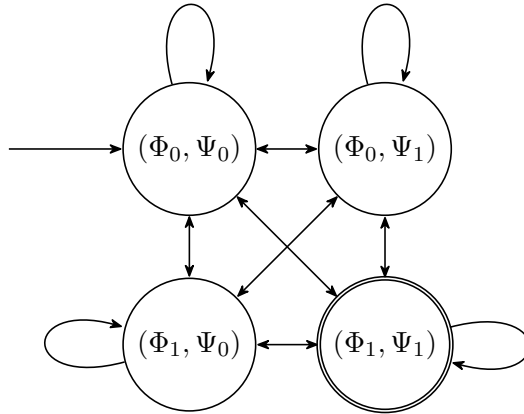


Figure 5: Acceptance probability $\phi\psi$

C651 It is straightforward to build a PFA whose acceptance probability is the product
 C652 $\phi(a)\psi(a)$, see Figure 5: This PFA simulates the two PFAs for v_1, \dots, v_k and for w_1, \dots, w_k
 C653 simultaneously and accepts if both PFAs accept. The resulting *product PFA* has four
 C654 states $\{\Phi_0, \Phi_1\} \times \{\Psi_0, \Psi_1\}$. Similarly, we can build a PFA with acceptance probability
 C655 $\phi(a)^2$: We simulate two *independent* copies of the PFA for v_1, \dots, v_k . This leads again to
 C656 four states. To get acceptance probability $1 - \phi(a)^2$, we complement the set of accepting
 C657 states. The PFA for $1 - \psi(a)^2$ follows the same principle. Finally, we mix the three PFAs
 C658 in the ratio $\frac{1}{2} : \frac{1}{4} : \frac{1}{4}$, as shown in Figure 6a.

C659 The dash-dotted arrows from the start state to three “local start states” inside the
 C660 square boxes denote random transitions that should be thought of as happening before
 C661 the algorithm reads its first input symbol. In the PFA, such a transition is actually
 C662 carried out in combination with the subsequent transition for the input symbol inside
 C663 one of the square boxes, as part of the transition out of the start state when reading the
 C664 first input symbol.

C665 More formally, if M_i is a 12×12 transition matrix for the 12 states in the square
 C666 boxes and $\pi_0^T = (\frac{1}{2}, 0, 0, 0, \frac{1}{4}, 0, 0, 0, \frac{1}{4}, 0, 0, 0)$ is the starting distribution, the 13×13
 C667 matrix that includes the start state as the first state is

$$C668 \quad \begin{pmatrix} 0 & \pi_0^T M_i \\ 0 & M_i \end{pmatrix}.$$

C669 ¹⁰Section 5.5.1 describes an alternative way to achieve the same effect.

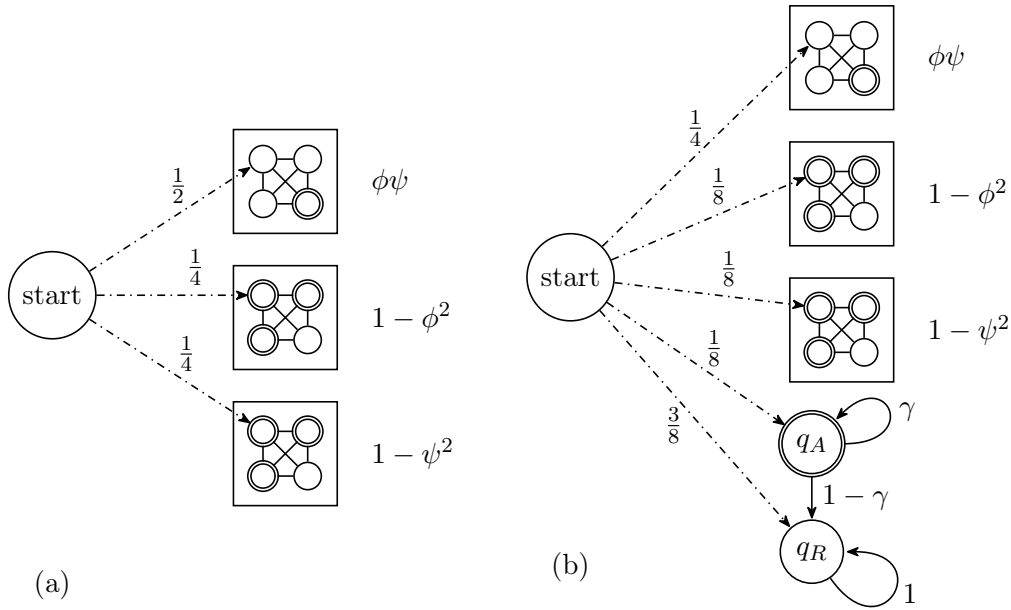


Figure 6: (a) Acceptance probability $\frac{1}{2} - \frac{1}{4}(\phi - \psi)^2$. (b) $\frac{1}{4} - \frac{1}{8}(\phi - \psi)^2 + \varepsilon$

C670 The introduction of the new start state has the beneficial side effect of eliminating
 C671 the empty word ϵ from the recognized language. The empty word would otherwise satisfy
 C672 the equation $\phi(a) = \psi(a)$, because $\phi(\epsilon) = \psi(\epsilon) = 0$.

C673 In total, we have now 13 states, 7 of which are accepting. As an intermediate unde-
 C674 cidability result, we can thus state:

C675 **Proposition 1.** *The following problem is undecidable:*

C676 *Given a finite set \mathcal{M} of stochastic matrices of size 13×13 with binary fractions as*
 C677 *entries, is there a product $M_1 M_2 \dots M_m$, with $M_i \in \mathcal{M}$ for all $i = 1, \dots, m$, such that*
 C678 *the sum of the 7 rightmost entries in the top row is $\geq \frac{1}{2}$?* \square

C679 5.4 Achieving strict inequality

C680 Proposition 1 almost describes a PFA, except that the convention for a PFA to recognize
 C681 a word is strict inequality ($> \lambda$). We thus have to raise the probability just a tiny bit,
 C682 without raising any of the values $< \lambda$ to become bigger than λ .

C683 Since all probabilities are rational, this can be done as follows, see Figure 6b. In our
 C684 case, all transition probabilities within the square boxes are multiples of some small unit

$$C685 \quad \gamma := 4^{-\max\{|v_i|, |w_i| : 1 \leq i \leq k\}}.$$

C686 The original PFA is entered with probability $1/2$. The transition probabilities from
 C687 the start state into the original PFA are now multiples of $\gamma/8$. (Remember that such a
 C688 transition consists of a transition from the start state along a dash-dotted arrow combined
 C689 with a transition inside a square box.) We create a new accepting state q_A that is
 C690 chosen initially with probability $1/8$. Whenever a symbol is read, the PFA stays in that
 C691 state with probability γ , and otherwise it moves to some absorbing state q_R . With the
 C692 remaining probability $3/8$, we go to q_R directly.

C693 The new part contributes $\varepsilon := \frac{1}{8}\gamma^{|a|}$ to the acceptance probability of every nonempty
 C694 word a . From the old part we have $\frac{1}{4} - \frac{1}{8}(\phi - \psi)^2$, and we know that this probability
 C695 is a multiple of $\frac{1}{8}\gamma^{|a|} = \varepsilon$. Thus, if this probability is less than $1/4$, it cannot become

C696 greater than $1/4$ by adding ε . If it was equal to $1/4$ (i.e., if a is a solution to the PCP),
 C697 it becomes greater than $1/4$.

C698 **Proposition 2.** *It is undecidable whether the language recognized by a PFA with 15*
 C699 *states with cutpoint $\lambda = 1/4$ is empty.* \square

C700 This PFA has a fixed start state.

C701 The cutpoint can be changed to any positive rational value less than $1/2$ by adjusting
 C702 the initial split probability between the original PFA of Figure 6a and the states q_A
 C703 and q_R . Cutpoints between $1/2$ and 1 can be achieved at the expense of adding another
 C704 accepting state.

C705 According to Neary [24], the PCP is already undecidable with as few as five word
 C706 pairs. Therefore, the size of the input alphabet in Proposition 2, or the number of
 C707 matrices \mathcal{M} in Proposition 1 can be restricted to 5.

C708 5.5 History of ideas

C709 The binary automaton (Section 5.1) and its generalization to other bases than 2 appears
 C710 already in Rabin’s 1963 paper [31], and it is credited to E. F. Moore. The basic m -ary
 C711 automaton processes single digits from $\{0, \dots, m - 1\}$. The binary automaton matrix
 C712 in Section 5.1 for variable-length input words u is the product of several such single-
 C713 digit matrices. Instead of binary automata, Nasu and Honda [23] use ternary (triadic)
 C714 automata with digits $\{0, 1, 2\}$, of which only $\{1, 2\}$ are used in order to avoid the trailing
 C715 zeros issue.

C716 The equality test for probabilities (constructing a PFA to accept words a with $\phi(a) =$
 C717 $\psi(a)$ from two PFAs with acceptance probabilities $\phi(a)$ and $\psi(a)$, Section 5.3), including
 C718 the method of adding a small probability to change $\geq \lambda$ into $> \lambda$ (Section 5.4) is given
 C719 in Nasu and Honda [23, Lemma 11, pp. 259–260]. The authors credit H. Matuura, Y.
 C720 Inagaki, and T. Hukumura for the key ideas (a technical report and a conference record,
 C721 both from 1968 and in Japanese) [23, p. 261].

C722 Claus already observed [7, p. 158, remark after the proof of Satz 28] that the con-
 C723 struction leads to a bounded number of states. The details have been worked out above.

C724 As I haven’t been able to survey the rich literature on probabilistic automata, I may
 C725 very well have overlooked some earlier roots of these ideas.

C726 Nasu and Honda [23], in a footnote to Theorem 21, their main result about the
 C727 undecidability of PFA Emptiness, write that “it reduces to a statement in p. 150” of
 C728 a paper of Marcel Schützenberger [37] from 1963¹¹ [23, footnote 6 on p. 270, referring
 C729 to the remark before Lemma 12, p. 261]. In that paper, Schützenberger derives some
 C730 undecidability results, using, among others, the PCP, but I am not able to see the
 C731 connection.

C732 Nasu and Honda prove the undecidability of two more questions in connection with
 C733 the language recognized by a PFA: whether the language is regular, or whether the
 C734 language is context-free [23, Theorem 22, p. 270], see Appendix A.1.

C735 5.5.1 Gimbert and Oualhadj 2010, following Bertoni 1975

C736 In 2010, Gimbert and Oualhadj [15], after joining the lamentations over the impenetra-
 C737 bility of Paz’s treatment [30] and the long and technical proof of Madani, Hanks, and
 C738 Condon [19] (see Section 4), presented “a new simple undecidability proof of the Empti-
 C739 ness Problem” as one of the results of their paper. I review this proof in greater detail in

¹¹<https://monge.univ-mlv.fr/~berstel/Mps/Travaux/A/A/1963-4ElementaryFamAutomataSymptAut.pdf>

C742 order to put it in the proper historical perspective. I will point where it differs from the
 C743 proof shown above, and I wish to correct an erroneous attribution of a crucial auxiliary
 C744 result to an early paper of Bertoni.

C745 As a first step, the proof starts with the translation of the PCP to the equality test
 C746 $\phi(a) = \psi(a)$, as in Sections 5.1–5.2.

C747 The treatment of the equality test in the second step is different from Nasu and
 C748 Honda’s (Section 5.3): The test $\phi(a) = \psi(a)$ is first reduced to the *Equality Problem*
 C749 $\frac{1}{2}\phi(a) + \frac{1}{2}(1 - \psi(a)) = \frac{1}{2}$, using complementation and convex combination of PFAs.¹²
 C750 This Equality Problem (“Is the acceptance probability of a given PFA *equal* to $\lambda = \frac{1}{2}$?”)
 C751 is reduced to the emptiness question with the $\geq \lambda$ criterion by the relation $x = \frac{1}{2} \iff$
 C752 $x(1 - x) \geq \frac{1}{4}$, and from there one can conclude that the Equality Problem is undecidable.
 C753

Putting everything together, this amounts to the formula

$$C754 \quad \left(\frac{1}{2}\phi + \frac{1}{2}(1 - \psi)\right)\left(\frac{1}{2}\psi + \frac{1}{2}(1 - \phi)\right) = \frac{1}{4} - \frac{1}{4}(\phi - \psi)^2,$$

C755 giving almost the same result as formula (8).¹³ I find this two-step approach concep-
 C756 tually simpler than the mixture of three automata in Section 5.3. If implemented in a
 C757 straightforward way, it would lead to 16 states.

C758 The third and final step adds a small probability to achieve strict inequality, in
 C759 essentially the same way as described in Section 5.4, using three additional states.

C760 Fijalkow, as part of a short 8-page note from 2017 [13, p. 15], has given a nice half-
 C761 page survey of this proof.¹⁴ As a complementary result, he also showed an alternative to
 C762 the binary automaton that has acceptance probability $0.u$ [13, Figure 1, p. 14] and that
 C763 satisfies a similar multiplication law as the binary automaton 5, see Figure 7. I find it
 C764 less mysterious than the binary automaton, but it requires 3 states.

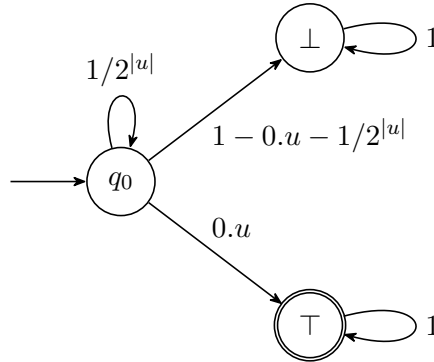


Figure 7: A simpler automaton for acceptance with a given binary probability $0.u$ when reading a particular symbol

C765 For the first one-and-a-half steps of the proof, leading to the claim that the Equality
 C766 Problem is undecidable [15, Proposition 1], Gimbert and Oualhadj credit a short paper
 C767 of Alberto Bertoni from 1975 [1].

C768 It is true that Bertoni [1, p. 111, item a)] *did* build a PFA with acceptance probability
 C769 $\frac{1}{2}\phi(a) + \frac{1}{2}(1 - \psi(a)) = \frac{1}{2} + (\phi(a) - \psi(a))/2$ (and with 4 states) as an intermediate step.
 C770 However, a statement that the Equality Problem is undecidable appears nowhere in

C771 ¹²See also Paz [30, Lemma 6.1 in Section IIIB, p. 183].

C772 ¹³The discrepancy between the constant $1/4$ in this formula and the claimed cutpoint $\lambda = 1/2$ [15,
 C773 Theorem 1] is not addressed in [15].

C774 ¹⁴However, Fijalkow shortcuts the third step, the transition from acceptance with $\geq \lambda$ to acceptance
 C775 with $> \lambda$, by appealing to the following invalid argument: Since the existence of input strings a with
 C776 acceptance probability $\phi(a) \geq 1/2$ is undecidable and the existence of input strings with $\phi(a) = 1/2$ (the
 C777 Equality Problem) is undecidable, the existence of input strings with $\phi(a) > 1/2$ is also undecidable.

C778 Bertonni [1]. In fact, the PCP instances of Bertonni are constructed in such a way that
 C779 they are guaranteed to have no (finite) solutions, and thus the Equality Problem would
 C780 always have the answer “no”.¹⁵

C781 Bertonni refers to Rabin [31] and Paz [30], but not to Nasu and Honda [23].

C782 Gimbert and Oualhadj, in an appendix of the technical report that is the full version
 C783 of [15],¹⁶ present a proof of the undecidability of the Equality Problem that is supposedly
 C784 due to Bertonni. This proof is essentially the proof that I have sketched above, using binary
 C785 automata and their mixture to generate the acceptance probability $\frac{1}{2}\phi(a) + \frac{1}{2}(1 - \psi(a))$,
 C786 except that Gimbert and Oualhadj (and following them, Fijalkow [13]), neglect to get rid
 C787 of the empty solution of the PCP.¹⁷ Fijalkow [13] adds to the mystifications by indirectly
 C788 attributing the undecidability of the *Emptiness Problem* to Bertonni [1].¹⁸

C789 5.6 Saving two states by merging indistinguishable states

C790 In the PFA with acceptance probability $\phi(a)^2$, where we simulate two independent copies
 C791 of the same PFA, we can see that the states (Φ_0, Ψ_1) and (Φ_1, Ψ_0) of Figure 5 become
 C792 indistinguishable when $\phi = \psi$. Thus, they can be merged into one state, denoted by
 C793 $\{\Phi_0, \Phi_1\}$, and we reduce the number of states by one, see Figure 8a–b.

C794 More precisely, if we denote the transition probabilities of the original binary automa-
 C795 ton by

$$C796 \quad B(u) = \begin{array}{c} \Phi_0 \quad \Phi_1 \\ \Phi_0 \left(\begin{array}{cc} p_{00} & p_{01} \\ p_{10} & p_{11} \end{array} \right), \\ \Phi_1 \end{array}$$

C797 the 3-state PFA has the following transition matrix:

$$C798 \quad \begin{array}{c} (\Phi_0, \Phi_0) \quad \{\Phi_0, \Phi_1\} \quad (\Phi_1, \Phi_1) \\ (\Phi_0, \Phi_0) \left(\begin{array}{ccc} p_{00}^2 & 2p_{00}p_{01} & p_{01}^2 \\ p_{00}p_{10} & p_{01}p_{10} + p_{00}p_{11} & p_{01}p_{11} \\ p_{10}^2 & 2p_{10}p_{11} & p_{11}^2 \end{array} \right) \end{array} \quad (9)$$

C799 When the reduced automaton is in the state $\{\Phi_0, \Phi_1\}$, we can think of the original 4-state
 C800 automaton being in one of the states (Φ_0, Φ_1) or (Φ_1, Φ_0) , each with probability $1/2$.

C801 For the PFAs in Propositions 1 and 2, the number of states can thus be reduced
 C802 by 2, as stated in the following proposition. Figure 8c illustrates the automaton for

C803 ¹⁵Bertonni’s goal was to prove that it is undecidable whether $1/2$ is an isolated cutpoint (see footnote 7).
 C804 For this, he constructed, for a given Turing machine T , a PCP with word pairs (v_i, w_i) with the following
 C805 property: The longest common prefixes of the strings $v_{a_1}v_{a_2}\dots v_{a_m}$ and $w_{a_1}w_{a_2}\dots w_{a_m}$ have bounded
 C806 length iff the Turing machine T halts [1, end of Section 2, p. 110]. (The first condition is equivalent to
 C807 $1/2$ being an isolated cutpoint of $\frac{1}{2} + (\phi(a) - \psi(a))/2$ [1, Theorem 3 of Section 3, p. 111].)

C808 The classic construction of a PCP that has a solution iff T halts (Section 6.1) would not work in this
 C809 context. Clearly, if T runs forever, there will be arbitrarily long prefixes. However, if T terminates and
 C810 the PCP has a solution, one can create arbitrarily long common prefixes simply by repeating the PCP
 C811 solution periodically and introducing a deviation at some point. Thus, $\lambda = 1/2$ would never be isolated.

C812 As an additional puzzle, the 2-state automaton that Bertonni specifies [1, Theorem 1 of Section 3,
 C813 p. 110] very roughly resembles a binary automaton. However, as written, it does not perform its claimed
 C814 function. I assume he must have meant the binary automaton. The k -ary automaton is written correctly
 C815 in a follow-up paper [2] that deals with further decidability questions about isolated cutpoints.

C816 ¹⁶https://hal.science/hal-00456538v3/file/gimbert_oualhadj_probabilistic_automata.pdf,
 C817 p. 13

C818 ¹⁷Also, the third step [15, Proposition 2] of the proof depends having a *simple* PFA, whose transition
 C819 probabilities are 0, $\frac{1}{2}$, or 1. The claim that their construction for the second step yields such a PFA [15,
 C820 Problem 2 and Proposition 1] is not substantiated.

C821 ¹⁸“[Gimbert and Oualhadj 2010] gave a simple exposition of the undecidability proof of Bertonni [Bertonni
 C822 1974] for the emptiness problem.” As for the date: December 1974 is the date of the conference whose
 C823 proceedings contain Bertonni’s paper [1] and were published in 1975.

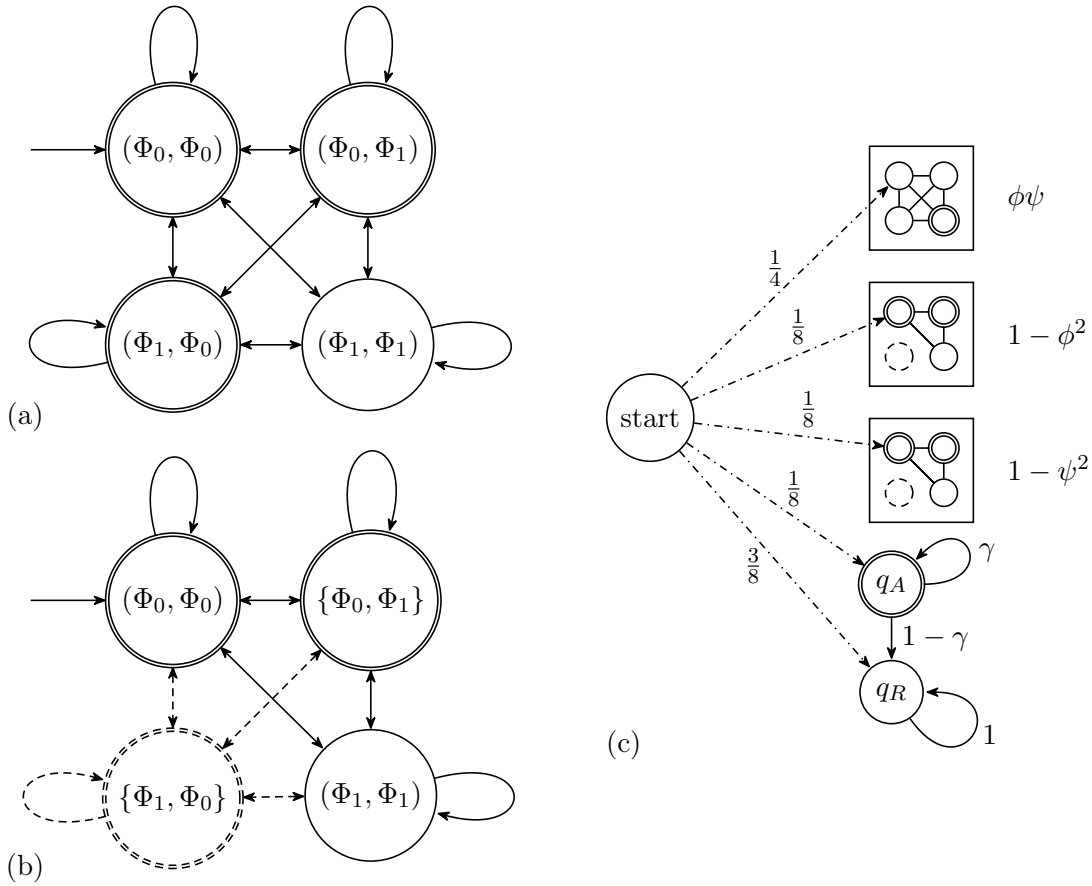


Figure 8: (a) Acceptance probability $1 - \phi^2$ with 4 states (b) with 3 states. (c) Acceptance probability $\frac{1}{4} - \frac{1}{8}(\phi - \psi)^2 + \varepsilon$ with 13 states

C824 Proposition 3b. We will show some explicit examples of transition matrices for this
 C825 automaton below, in Section 6.5.

C826 **Proposition 3.**

C827 (a) *The following problem is undecidable:*

C828 *Given a finite set \mathcal{M} of stochastic matrices of size 11×11 with binary fractions as*
 C829 *entries, is there a product $M_1 M_2 \dots M_m$, with $M_i \in \mathcal{M}$ for all $i = 1, \dots, m$, such*
 C830 *that the sum of the 5 rightmost entries in the top row is $\geq \frac{1}{2}$?*

C831 (b) *It is undecidable whether the language recognized by a PFA with 13 states with cut-*
 C832 *point $\lambda = 1/4$ is empty. \square*

C833 **5.7 Saving the start state by using the Modified Post Correspondence Problem**

C834 We can eliminate the start state by using the *Modified Post Correspondence Problem*
 C835 (MPCP). It differs from the PCP in one detail: The pair (v_1, w_1) must be used as the
 C836 *starting pair*, and it cannot be used in any other place. In other words, the solution must
 C837 satisfy the constraints $a_1 = 1$, and $a_i > 1$ for $i = 2, \dots, m$. The MPCP is often used
 C838 as an intermediate problem when reducing the Halting Problem for Turing machines to
 C839 the PCP, and then it takes some extra effort to reduce the MPCP to the PCP, see for

C840 example [18, Lemma 8.5] or [38, p. 189]. In our situation, the MPCP is actually the more
C841 convenient version of the problem.

C842 The idea is to apply the transition for the first letter a_1 right away, and use the
C843 resulting distribution on the states as the starting distribution π .

C844 There is still a small technical discrepancy: In the formulas (6) and (7) for the
C845 acceptance probability, the first letter of the sequence a determines the *last* pair of words
C846 to be concatenated. Thus we must reverse all words v_i and w_i and turn the MPCP into
C847 a *reversed* MPCP, where the *last* pair in the concatenation is prescribed to be the pair
C848 (v_1, w_1) :

C849 The Reversed Modified Post Correspondence Problem (RMPCP).

C850 We are given a list of pairs of words $(v_1, w_1), (v_2, w_2), \dots, (v_k, w_k)$ over the
C851 alphabet $\{0, 1\}$ such that v_1 and w_1 end with 1. The problem is to decide if
C852 there is a sequence a_2, \dots, a_m of indices $a_i \in \{2, \dots, k\}$ such that

$$C853 \quad v_{a_m} v_{a_{m-1}} \dots v_{a_2} v_1 = w_{a_m} w_{a_{m-1}} \dots w_{a_2} w_1 .$$

C854 This is of course just a trivial variation of the MPCP. The translation of (6) and (7) can
C855 now be applied directly. Moreover, the trailing zeros issue disappears, since v_1 and w_1
C856 end with 1. This extra condition can be easily fulfilled by appending a 1 to v_1 and w_1 if
C857 necessary.

C858 **Proposition 4.** *It is undecidable whether the language recognized by a PFA with 12*
C859 *states with cutpoint $\lambda = 1/4$ is empty.*

C860 *Proof.* The above construction that has led to Proposition 3b gives a set of 13×13
C861 matrices such that the index sequence $a_1 a_2 \dots a_m$ is a solution of the PCP if and only if

$$C862 \quad e_1^T M_{a_1} M_{a_2} \dots M_{a_{m-1}} M_{a_m} f > \frac{1}{4},$$

C863 where e_1 is the first unit vector $(1, 0, \dots, 0)$ and f is a vector with 6 zeros and 7 ones.
C864 For every other index sequence, the value of the expression is $< \frac{1}{4}$.

C865 For the reversed MPCP the first matrix $M_{a_1} = M_1$ is specified. Thus the product
C866 $e_1^T M_1$ has a fixed value π^T , and we can replace it by this vector:

$$C867 \quad \pi^T M_{a_2} \dots M_{a_{m-1}} M_{a_m} f$$

C868 This is the expression for the acceptance probability starting from an initial probability
C869 distribution π . The remaining matrix product is not allowed to use M_1 , and this is easily
C870 ensured by removing M_1 from \mathcal{M} .

C871 The original PFA goes from the start state to the 12 other states and never returns to
C872 the start state; thus we can eliminate the start state and only use the 12×12 submatrices
C873 for the remaining states. \square

C874 We mention that with cutpoint $\frac{1}{2}$ and the weak inequality $\geq \frac{1}{2}$ as acceptance criterion
C875 instead of $> \frac{1}{4}$, we don't need the extra states q_A and q_R , and the number of states is
C876 reduced to 10.

C877 6 Fixing the set of matrices by using a universal Turing machine

C878 We can achieve stronger and more specific results by tracing back the undecidability
C879 of the PCP to the Halting Problem. In particular, we will look at a universal Turing

C904 starting distribution π , whereas the transition matrices M_i depend only on the rules of
 C905 the Turing machine, which, for a universal Turing machine, are fixed!

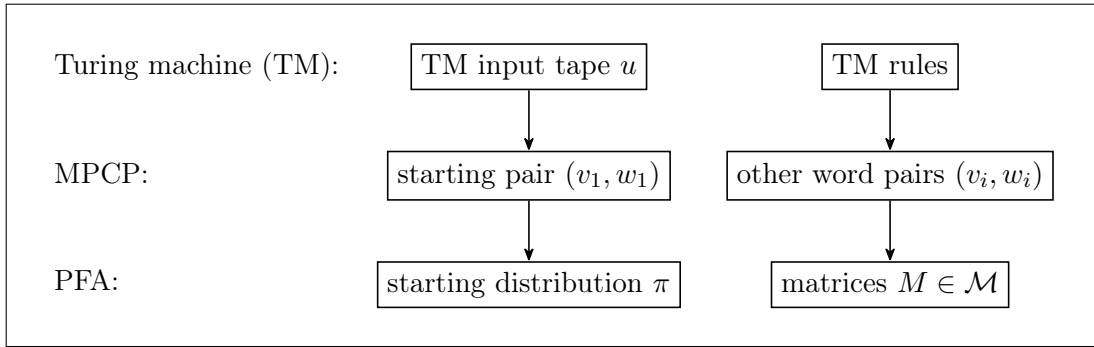


Figure 10: How the PFA is constructed from a Turing machine via an MPCP

C906 6.2 List of word pairs of the MPCP

C907 Since we want a MPCP with as few pairs as possible, we review the construction of the
 C908 MPCP from the Turing machine in detail. We follow the construction from Sipser [38,
 C909 Section 5.2, Part 5, p. 187] to ensure that the configurations are padded with sufficiently
 C910 many blank symbols. This eliminates the need to deal with special cases when the Turing
 C911 machine reaches the “boundary” of the tape in the representation as a finite string.

C912 Let Γ denote the tape alphabet including the blank symbol \sqcup , and let Q denote the
 C913 set of states of the Turing machine. The words v_i and w_i of the MPCP use the alphabet
 C914 $\Gamma \cup Q \cup \{\#, H\}$ with two extra symbols: a separation symbol $\#$ and a halting symbol H .

- C915 • If the input string for the Turing machine is $u \in \{0, 1\}^*$, we define the *starting pair*
 C916 $(v_1, w_1) = (\#, \#_q u \#)$, where q_0 is the start state of the Turing machine.

C917 The other pairs (v_i, w_i) are as follows:

- C918 • For *copying* from the shorter string to the more advanced string, we have the pairs
 C919 (s, s) for all $s \in \Gamma$.
- C920 • We have another copying pair $(\#, \#)$, and the *padding pair* $(\#, \sqcup \#)$. We are allowed
 C921 to (nondeterministically) emit an additional blank symbol at both ends of the
 C922 configuration.
- C923 • For each *right-moving rule* (q, s, s', R, q') , the pair $(qs, s'q')$. (Such a rule means that
 C924 when the Turing machine is in state q and reads the tape symbol s , it overwrites
 C925 the symbol s with s' , moves one step to the right on the tape, and changes to
 C926 state q' .)
- C927 • For each *left-moving rule* (q, s, s', L, q') and for each $t \in \Gamma$, the pair $(tqs, q'ts')$.
- C928 • For each *halting rule* $(q, s, -)$, the pair (qs, H) . The character H represents the
 C929 fact that the machine has halted.
- C930 • For each $s \in \Gamma$, the *erasing pairs* (Hs, H) and (sH, H) . The halting symbol absorbs
 C931 all symbols on the tape one by one.
- C932 • Finally, the *finishing pair* $(H\#\#, \#)$. This is the only way how the two strings can
 C933 come to a common end.

C934 In total, these are $3|\Gamma| + 3$ pairs, plus $|\Gamma|$ pairs for each left-moving rule, plus one pair
 C935 for each right-moving or halting rule, plus the starting pair.

C936 We can save copying rules by encoding everything in binary, for example using the
 C937 codewords $\mathbf{bab}, \mathbf{baab}, \mathbf{baaab}, \dots$ over the alphabet $\{\mathbf{a}, \mathbf{b}\}$. No codeword is a substring of
 C938 another codeword, and the strings that are composed from such codewords can always
 C939 be uniquely decoded. Then we can replace the $|\Gamma| + 1$ copying pairs for $\Gamma \cup \{\#\}$ by just
 C940 two pairs (\mathbf{a}, \mathbf{a}) and (\mathbf{b}, \mathbf{b}) .

C941 This leads to a baseline of $2|\Gamma| + 4$ pairs instead of $3|\Gamma| + 3$.

C942 The Turing machine $U_{15,2}$ that we will use in the next section, has only $|\Gamma| = 2$
 C943 symbols. In this case, we would save one pair. Thus, the number of matrices in the
 C944 theorems of this section and Section 7 can in fact be reduced by one. We have not
 C945 investigated the bit size of the transition matrices that we would get from this approach.

C946 6.3 Using a universal Turing machine

C947 We have looked at the parameters of various universal Turing machines in the literature
 C948 in order to see which ones give the smallest number of pairs (v_i, w_i) for our PCP. The
 C949 best result is obtained from the machine $U_{15,2}$ of Neary and Woods [26, Section 3.5]¹⁹
 C950 Its tape alphabet, including the blank symbol, has size $|\Gamma| = 2$. It has 15 states, not
 C951 counting the halting state. It has 15 left-moving rules, 14 right-moving rules, and 1
 C952 halting rule. In terms of PCP pairs, left-moving rules are more costly than right-moving
 C953 rules, but we have the freedom to swap left-moving with right-moving rules by flipping
 C954 the Turing machine's tape. We have to switch to the nonstandard convention of starting
 C955 the Turing machine over the rightmost input character, but this is easily accomplished
 C956 in the construction of the starting pair (v_1, w_1) . Thus, with 14 left-moving rules and 16
 C957 right-moving and halting rules, we get $3 \times 2 + 3 + 14 \times 2 + 16 = 53$ pairs, plus the starting
 C958 pair (v_1, w_1) that encodes the input.

C959 In some sense, this can be regarded as a *universal* MPCP: all pairs except the starting
 C960 pair are fixed.

C961 We can now establish a weaker version of Theorem 2b, with matrices of dimension
 C962 12×12 instead of 11×11 .

C963 **Proposition 5.** *There is a fixed set of 53 stochastic matrices \mathcal{M}'''' of dimension 12×12 ,
 C964 whose entries are multiples of 2^{-22} , and a fixed 0-1-vector $f \in \{0, 1\}^{12}$, for which the
 C965 following question is undecidable:*

C966 *Given a probability distribution $\pi \in \mathbb{Q}^{12}$ whose entries are binary fractions, is there
 C967 a product $M_1 M_2 \dots M_m$, with $M_i \in \mathcal{M}''''$ for all $i = 1, \dots, m$, such that*

$$C968 \quad \pi^T M_1 M_2 \dots M_m f > \frac{1}{4} ?$$

C969 *In other words, is the language recognized by the PFA with starting distribution π and
 C970 cutpoint $\lambda = \frac{1}{4}$ nonempty?*

C971 *Proof.* We specialize the proof of Proposition 4 to the current setting. The important
 C972 point, as discussed above and shown in Figure 10, is that the matrices in \mathcal{M} depend only
 C973 on the word pairs that reflect the rules of the universal Turing machine $U_{15,2}$, which are
 C974 fixed, and we have already calculated that there are 53 of these matrices.

C975 We must not forget that the symbols of the alphabet $\Gamma \cup Q \cup \{\#, H\}$, in which the
 C976 word pairs (v_i, w_i) of the MPCP are written, have to be encoded somehow into the binary
 C977 alphabet $\{0, 1\}$ in order to define the matrices of the PFA, and we have to ensure that
 C978 the codes of v_1 and w_1 end with 1, for example by letting the code for $\#$ end with 1.

C979 ¹⁹see also <http://mural.maynoothuniversity.ie/12416/>, with incorrect page numbers, however

C980 There is one technicality that needs to be resolved. The quantity γ was required to be
C981 a common divisor of the matrix entries, and it depends on the maximum lengths $|v_i|$ and
C982 $|w_i|$ of the input words. However, the words v_1 and w_1 depend on the input tape, and
C983 thus, their lengths $|v_1|$ and $|w_1|$ cannot be bounded in advance. (The remaining word
C984 pairs depend only on the Turing machine.) The solution is to carry out the imagined
C985 first transition (which is not encoded into a transition matrix in \mathcal{M} , but determines the
C986 starting distribution π) with a sufficiently small value of γ , namely $\gamma_1 = 4^{-\max\{|v_1|, |w_1|\}}$,
C987 where the lengths $|v_1|$ and $|w_1|$ are measured in the binary encoding. The other transitions
C988 from the state q_A can be carried out with the fixed value γ that is sufficient for those
C989 entries. Table 2 shows the starting distribution π resulting from this construction. Since
C990 π is allowed to depend on the input, we have solved the problem.

state q	π_q	state q	π_q	state q	π_q
(Φ_0, Ψ_0)	$\frac{1}{4}(1 - 0.v_1)(1 - 0.w_1)$	(Φ_0, Φ_0)	$\frac{1}{8}(1 - 0.v_1)^2$	$\{\Psi_0, \Psi_1\}$	$\frac{1}{4}(1 - 0.w_1)0.w_1$
(Φ_0, Ψ_1)	$\frac{1}{4}(1 - 0.v_1)0.w_1$	$\{\Phi_0, \Phi_1\}$	$\frac{1}{4}(1 - 0.v_1)0.v_1$	(Ψ_1, Ψ_1)	$\frac{1}{8}(0.w_1)^2$
(Φ_1, Ψ_0)	$\frac{1}{4} \cdot 0.v_1(1 - 0.w_1)$	(Φ_1, Φ_1)	$\frac{1}{8}(0.v_1)^2$	q_A	$\frac{1}{8}\gamma_1$
(Φ_1, Ψ_1)	$\frac{1}{4} \cdot 0.v_1 \cdot 0.w_1$	(Ψ_0, Ψ_0)	$\frac{1}{8}(1 - 0.w_1)^2$	q_R	$\frac{1}{2} - \frac{1}{8}\gamma_1$

Table 2: Starting probabilities π for Proposition 5

C991 We have now established the existence of 53 fixed matrices \mathcal{M}'''' and a finishing
C992 0-1-vector f for which the decision problem of Proposition 5 is undecidable.

C993 6.4 An efficient code

C994 In order to say something about the entries of these matrices, we have to be more specific
C995 about the way how the alphabet $\Gamma \cup Q \cup \{\#, H\}$ is encoded. The words v_i and w_i that
C996 come from the Turing machine rules are actually quite short: they have at most 3 letters.
C997 More precisely, they consist of at most one “state” symbol from $Q \cup \{H\}$, plus at most
C998 two letters from the tape alphabet $\Gamma \cup \{\#\}$. The Turing machine $U_{15,2}$ has $|Q| = 15$ states
C999 and a tape alphabet of size $|\Gamma| = 2$.

C1000 In this situation, a variable-length code is more efficient than a fixed-length code. We
C1001 can use 5-letter codes of the form $0****$ for the 15 states plus the halting state H . This
C1002 leaves the 3-letter codes $1**$ for the 3 symbols $\Gamma \cup \{\#\}$, leading to word lengths bounded
C1003 by $5 + 3 + 3 = 11$. In the binary automaton, the transition probabilities are therefore
C1004 multiples of 2^{-11} . Since each box carries out two binary automata simultaneously, the
C1005 transition probabilities are multiples of 4^{-11} . \square

C1006 With a weak inequality like $\geq \frac{1}{2}$ instead of $> \frac{1}{4}$ as acceptance criterion, we don’t need
C1007 the extra states q_A and q_R , and the size of the matrices for which Proposition 5 holds
C1008 can be reduced to 10×10 .

C1009 As mentioned after Proposition 2, the cutpoint can be changed to a different value;
C1010 then the constraint that the input distribution π consists of binary fractions must be
C1011 abandoned. Since the change only affects the very first transition, the fixed matrix set
C1012 \mathcal{M} remains unchanged.

C1013 The above variable-length code seems to be pretty efficient, but it wastes one of the
C1014 four codewords $1**$. By looking at the actual rules of the machine $U_{15,2}$ and fiddling
C1015 with the code, it might be possible to improve the power 22 in the denominator of the
C1016 binary fractions.

6.5 Example matrices

C1017

C1018

C1019

C1020

For illustration, we compute some matrices of the set \mathcal{M}''' explicitly. We use the binary code $\# \doteq 101$, $\sqcup \doteq 100$. The copying pair $(\sqcup, \sqcup) \doteq (100, 100)$ is then translated into the block diagonal matrix

$$\overline{M}_{(\sqcup, \sqcup)} = \begin{pmatrix} \frac{1}{64} \begin{pmatrix} 16 & 16 & 16 & 16 \\ 12 & 20 & 12 & 20 \\ 12 & 12 & 20 & 20 \\ 9 & 15 & 15 & 25 \end{pmatrix} & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{64} \begin{pmatrix} 16 & 32 & 16 \\ 12 & 32 & 20 \\ 9 & 30 & 25 \end{pmatrix} & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{64} \begin{pmatrix} 16 & 32 & 16 \\ 12 & 32 & 20 \\ 9 & 30 & 25 \end{pmatrix} & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{2^{22}} & 1 - \frac{1}{2^{22}} \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix},$$

C1021

C1022

C1023

C1024

where the rows and columns correspond to the states as they are ordered in Table 2.²⁰ We use a bar in the notation $\overline{M}_{(\sqcup, \sqcup)}$ to remind us of the fact that we are dealing with the *reversed* MPCP, and therefore the words should be reversed (which, in this case, has no effect because we have only one-letter words).

C1025

C1026

C1027

Let us look at the erasing pair $(\#, \#)$. Here the reversal does have an effect, and the strings are actually $(\sqcup\#, \#) \doteq (100101, 101)$. (The codewords don't have to be reversed.) With these data, the matrix $\overline{M}_{(\#, \#)}$ looks as follows:

$$\begin{pmatrix} \frac{1}{512} \begin{pmatrix} 81 & 135 & 111 & 185 \\ 54 & 162 & 74 & 222 \\ 78 & 130 & 114 & 190 \\ 52 & 156 & 76 & 228 \end{pmatrix} & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{4096} \begin{pmatrix} 729 & 1998 & 1369 \\ 702 & 1988 & 1406 \\ 676 & 1976 & 1444 \end{pmatrix} & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{64} \begin{pmatrix} 9 & 30 & 25 \\ 6 & 28 & 30 \\ 4 & 24 & 36 \end{pmatrix} & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{2^{22}} & 1 - \frac{1}{2^{22}} \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

C1028

C1029

C1030

C1031

C1032

C1033

Finally, as our most elaborate example, we consider a left-moving rule of the Turing machine $U_{15,2}$ from [26]: $(q_9, \sqcup, \sqcup, L, q_1)$. This was originally a right-moving rule, but has been converted into a left-moving rule by flipping the tape. It produces two word pairs, since $|\Gamma| = 2$. One of these pairs is $(\mathbf{b}q_9\sqcup, q_1\mathbf{b}\sqcup)$, where \mathbf{b} is the other letter of the tape alphabet besides \sqcup . Coding this letter as $\mathbf{b} \doteq 110$ and the states in the most straightforward way as $q_1 \doteq 00001$ and $q_9 \doteq 01001$, we get, after reversal, the binary word

C1034

C1035

C1036

²⁰Since $v_i = w_i = \sqcup$ in this case, we have a chance to compare the straightforward 4×4 construction of the probability ϕ^2 (the upper left block) with the condensed representation with 3 states, in the two middle blocks.

C1037 pair $(\sqcup q_9 \mathbf{b}, \sqcup \mathbf{b} q_1) \doteq (100\ 01001\ 110, 100\ 110\ 00001)$ and the following transition matrix:

$$\begin{pmatrix} \frac{1}{2^{22}} \begin{pmatrix} 786126 & 1151282 & 915762 & 1341134 \\ 785180 & 1152228 & 914660 & 1342236 \\ 785295 & 1150065 & 916593 & 1342351 \\ 784350 & 1151010 & 915490 & 1343454 \end{pmatrix} & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{2^{22}} \begin{pmatrix} 894916 & 2084984 & 1214404 \\ 893970 & 2084828 & 1215506 \\ 893025 & 2084670 & 1216609 \end{pmatrix} & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{2^{22}} \begin{pmatrix} 690561 & 2022654 & 1481089 \\ 689730 & 2022268 & 1482306 \\ 688900 & 2021880 & 1483524 \end{pmatrix} & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{2^{22}} & 1 - \frac{1}{2^{22}} \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

C1038

7 Output values instead of a set of accepting states

C1039

In the expression for the acceptance probability in (1), π and f appear in symmetric roles. We will now fix the starting distribution π , and in exchange, we allow more general values f_q .

C1040

C1041

C1042

In the classic model of a PFA, f is a 0-1-vector: Once the input has been read and all probabilistic transitions have been made, acceptance is a yes/no decision. The state that has been reached is either accepting or not.

C1043

C1044

C1045

C1046

C1047

C1048

C1049

C1050

C1051

C1052

We can think of a general value f_q as a probability in a final acceptance decision, after the input has been read. Another possibility is that f_q represents a *prize* or *value* that that is gained when the process stops in state q , as in game theory. Then f_q does not need to be restricted to the interval $[0, 1]$. In this view, instead of the acceptance probability, we compute the *expected* gain (or loss) of the automaton. Following Carl Page [28], who was the first to consider this generalization, we call f the *output vector* and f_q the *output values*. Mathematically, it make sense to take the outputs even from some (complex) vector space (quantum automata?).

C1053

C1054

C1055

C1056

C1057

C1058

C1059

In our results, the values f_q are restricted to $[0, 1]$, and in fact, they have an interpretation as probabilities.

Turakainen [40, 41] considered the most general setting, allowing arbitrary positive or negative entries also for the matrices $M \in \mathcal{M}$ and the vectors π and f . He showed that the condition (1) with these more general data does not define a more general class of languages than a classic PFA, see also [7, §3.3.2, pp. 120–126] or [30, Proposition 1.1 in Section IIIB, p. 153].

C1060

7.1 Saving one more state by maintaining four binary variables

C1061

C1062

C1063

C1064

C1065

C1066

C1067

C1068

C1069

C1070

C1071

C1072

The PFA of Figure 8c mixes the PFAs for the three terms $\phi\psi$, $1 - \phi^2$, and $1 - \psi^2$ by deciding *in advance* which sub-automaton they should enter. As an alternative approach when arbitrary output values f_q are allowed, we can delay this decision to the end, when we decide whether to (probabilistically) accept the input, and this will allow us to further reduce the number of states by one.

The idea is to maintain four independent binary state variables $\Phi', \Phi'', \Psi', \Psi''$ throughout the process. Such a pool of variables is sufficient for any of the terms $\phi\psi$, $1 - \phi^2$, and $1 - \psi^2$. This would normally require $2^4 = 16$ states. As discussed above, the combinations (Φ'_0, Φ''_1) and (Φ'_1, Φ''_0) need not be distinguished and can be merged into one state, denoted by $\{\Phi_0, \Phi_1\}$, and similarly for the Ψ variables. Thus, the overall number of states is reduced from 16 to $3 \times 3 = 9$ combinations q , one less than the 10 states in the three square boxes of Figure 8c.

C1073 As we will see, we have to set the nine entries \hat{f}_q of the output vector to the following
C1074 values.

$$\begin{array}{c|ccc}
 & (\Psi'_0, \Psi''_0) & \{\Psi_0, \Psi_1\} & (\Psi'_1, \Psi''_1) \\
 \hline
 (\Phi'_0, \Phi''_0) & 1/2 & 1/2 & 1/4 \\
 \{\Phi_0, \Phi_1\} & 1/2 & 5/8 & 1/2 \\
 (\Phi'_1, \Phi''_1) & 1/4 & 1/2 & 1/2
 \end{array} \tag{10}$$

C1075 Beware that this is an output *vector* $\hat{f}_q \in \mathbb{Q}^9$, which has been arranged in 3×3 tabular
C1076 form only for convenience. These output values result from the contributions to the three
C1077 terms $\frac{1}{2}\phi\psi$, $\frac{1}{4}(1 - \phi^2)$, $\frac{1}{4}(1 - \psi^2)$ of the overall acceptance probability as shown below,
C1078 where the states are arranged in the same matrix form as in (10):

$$\frac{1}{2} \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1/4 & 1/2 \\ 0 & 1/2 & 1 \end{pmatrix} + \frac{1}{4} \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 0 & 0 & 0 \end{pmatrix} + \frac{1}{4} \begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 0 \end{pmatrix} = \begin{pmatrix} 1/2 & 1/2 & 1/4 \\ 1/2 & 5/8 & 1/2 \\ 1/4 & 1/2 & 1/2 \end{pmatrix}$$

C1080 The fractional values in the first matrix appear for the following reason. We have reduced
C1081 the states for generating the acceptance probability ϕ^2 from 4 to 3 by merging two states
C1082 into one. Thus, when the PFA is, for example, in the state $(\{\Phi_0, \Phi_1\}, (\Psi'_1, \Psi''_1))$, it is
C1083 “really” in one of the two states $(\Phi'_0, \Phi''_1, \Psi'_1, \Psi''_1)$ or $(\Phi'_1, \Phi''_0, \Psi'_1, \Psi''_1)$, each with a share of
C1084 50%. If we consider the product $\phi\psi$ as built, say, from the conjunction (Φ'_1, Ψ'_1) , ignoring
C1085 the variables Φ'' and Ψ'' , only the second of these two states should lead to acceptance,
C1086 and therefore we get the fractional output value $1/2$.

C1087 We can change the cutpoint (for the original automaton, without the extra states q_A
C1088 and q_R) from $\lambda = 1/2$ to any rational value λ strictly between 0 and 1 by modifying
C1089 the output values \hat{f}_q in (10): By scaling both \hat{f} and λ down by the same factor, λ can
C1090 be brought arbitrarily close to 0. On the other hand, by applying the transformation
C1091 $x \mapsto 1 - \alpha(1 - x)$ for some constant $0 < \alpha \leq 1$ to \hat{f} and λ , the cutpoint λ can be moved
C1092 arbitrarily close to 1 [30, Proposition 1.4 of Section IIIB, p. 153].

C1093 7.2 Making all transition probabilities positive

C1094 By using an appropriate binary code, we can ensure that all transition matrices are
C1095 strictly positive. Rabin calls such PFAs *actual automata* and studies their properties [31,
C1096 Sections IX–XII, p. 242–245], see also [7, §3.2.3, pp. 115–118].

C1097 One can easily check that the transition matrix $B(u)$ for the binary automaton is
C1098 positive except when the string u consists only of zeros or only of ones. With only 3
C1099 symbols $\Gamma \cup \{\#\}$ using the 4 codewords $\mathbf{1}^{**}$, we can avoid the all-ones codeword $\mathbf{111}$ (as
C1100 in the code used for the examples in Section 6.5).

C1101 A state symbol other than H never appears alone in a word v_i or w_i . Thus, we can use
C1102 the codeword $\mathbf{00000}$ for one of the original states, and thereby ensure that the transition
C1103 matrices $B(v_i)$ and $B(w_i)$ are always positive. As discussed earlier, the encoded words
C1104 u_i and v_i have at most 11 bits, and hence the matrix entries are multiples of 2^{-11} . The
C1105 entries of the 3×3 transition matrix (9) are sums and products of entries of the 2×2
C1106 matrices $B(v_i)$ or $B(w_i)$, respectively, and are therefore positive multiples of 2^{-22} . Each
C1107 entry of the 9×9 transition matrix is obtained by multiplying appropriate entries of the
C1108 two 3×3 matrices, and is hence a positive multiple of 2^{-44} . (To say it more concisely,
C1109 the matrix is the Kronecker product, or tensor product, of the two 3×3 matrices.)

C1110 More generally, the entries are multiples of $\gamma^2 = 16^{-\max\{|v_i|, |w_i|; 1 \leq i \leq k\}}$.

C1111 **7.3 Fixing everything except the output vector, proof of Theorem 4**

C1112 We will from now on use superscripts like M^i or $M^{(v_i, w_i)}$ or $\overline{M}^{(v_i, w_i)}$ for the matrices that
 C1113 are associated to the word pairs (v_i, w_i) , in order to distinguish them from the notation
 C1114 M_j in the theorem below, where they are numbered in the order in which they are used
 C1115 in the matrix product of the solution.

C1116 For the version with fixed starting distribution, we use the original (unreversed)
 C1117 MPCP, where the *first* word pair in the solution, and hence the *last* matrix in the matrix
 C1118 product, is fixed.

C1119 We can save a matrix by observing that the *last* word pair in the PCP is also known:
 C1120 It is the finishing pair $(H\#\#, \#)$, and like the starting pair, this pair is used nowhere else.
 C1121 (This is the only pair, besides the starting pair, that has a different number of $\#$'s in the
 C1122 two components, and it is the only possibility how the string $v_1 v_{a_2} \dots v_{a_n}$ can catch up
 C1123 with the string $w_1 w_{a_2} \dots w_{a_n}$.)

C1124 For clarity, we formulate the (unreversed) Doubly-Modified Post Correspondence
 C1125 Problem (2MPCP), with two special pairs: a starting pair (v_1, w_1) and a finishing pair
 C1126 (v_2, w_2) :

C1127 We are given a list of pairs of words $(v_1, w_1), (v_2, w_2), \dots, (v_k, w_k)$ over the
 C1128 alphabet $\{0, 1\}$ such that v_2 and w_2 end with a 1. The problem is to decide
 C1129 if there is a sequence a_2, \dots, a_{m-1} of indices $a_i \in \{3, \dots, k\}$ such that

C1130
$$v_1 v_{a_2} v_{a_3} \dots v_{a_{m-1}} v_2 = w_1 w_{a_2} w_{a_3} \dots w_{a_{m-1}} w_2 .$$

C1131 The PFA starts deterministically in the state $(\Phi'_0, \Phi''_0, \Psi'_0, \Psi''_0)$. Thus, the 2MPCP has a
 C1132 solution if and only if the following inequality can be solved:

C1133
$$e_1^T M^2 M^{a_{m-1}} \dots M^{a_2} M^1 \hat{f} \geq \frac{1}{2}, \quad (11)$$

C1134 where \hat{f} is the output vector defined in (10). The matrix $M^2 = M^{(H\#\#, \#)}$ comes from the
 C1135 finishing pair $(H\#\#, \#)$ and is fixed, and M^1 depends on the input tape u of the Turing
 C1136 machine. With the substitutions

C1137
$$\pi^T := e_1^T M^2,$$

 C1138
$$f := M^1 \hat{f},$$

C1139 we can remove M^2 from the set of matrices \mathcal{M} , and this directly leads to part (a) of the
 C1140 following theorem:

C1141 **Theorem 4.**

C1142 (a) *There is a fixed set \mathcal{M}''' of 52 positive stochastic matrices of size 9×9 and a fixed*
 C1143 *starting distribution π , all with positive entries that are multiples of $1/2^{44}$, for which*
 C1144 *the following question is undecidable:*

C1145 *Given a vector $f \in \mathbb{Q}^9$ whose entries are binary fractions from the interval $[\frac{1}{4}, \frac{5}{8}]$, is*
 C1146 *there a product $M_1 M_2 \dots M_m$, with $M_j \in \mathcal{M}'''$ for all $j = 1, \dots, m$, with*

C1147
$$\pi^T M_1 M_2 \dots M_m f \geq \frac{1}{2} ?$$

C1148 (b) *There is a fixed set \mathcal{M}'' of 52 stochastic matrices of size 11×11 and a fixed starting*
 C1149 *distribution π , all of whose entries are multiples of $1/2^{45}$, for which the following*
 C1150 *question is undecidable:*

C1151 *Given a vector $f \in \mathbb{Q}^{11}$ whose entries are binary fractions from the interval $[0, 1]$, is*
 C1152 *there a product $M_1 M_2 \dots M_m$, with $M_j \in \mathcal{M}''$ for all $j = 1, \dots, m$, such that*

C1153
$$\pi^T M_1 M_2 \dots M_m f > \frac{1}{4} ?$$

C1154 *Proof.* For part (a), everything has already been said except for observing that the entries
C1155 of $f = M^1 \hat{f}$ are in the interval $[\frac{1}{4}, \frac{5}{8}]$ because M^1 is a stochastic matrix and the entries
C1156 of \hat{f} are in that interval.

C1157 For part (b), we add the same two states q_A and q_R as in Figure 6b (p. 18) and
C1158 Figure 8c, with $\gamma = 2^{-44}$. Initially, we select the original start state $(\Phi'_0, \Phi''_0, \Psi'_0, \Psi''_0)$ and
C1159 the state q_A each with probability $\frac{1}{2}$. Denoting by π_0 the corresponding vector with two
C1160 $\frac{1}{2}$ entries, the initial distribution π is then defined as

$$C1161 \quad \pi_0^T M^2 =: \pi^T, \quad (12)$$

C1162 and its entries are multiples of $\frac{1}{2^{45}}$.

C1163 The matrix M^1 is constructed from the starting pair (v_1, w_1) , and it uses the value

$$C1164 \quad \gamma_1 = 1/16^{\max\{|v_1|, |w_1|\}}, \quad (13)$$

C1165 where $|v_1|$ and $|w_1|$ are the lengths after the binary encoding.

C1166 The output values of the extra states are defined as $\hat{f}_{q_A} = 1/8$ and $\hat{f}_{q_R} = 0$. Since the
C1167 remaining output values in \hat{f} are multiples of $1/8$, the value $\hat{f}_{q_A} = 1/8$ is small enough
C1168 to ensure that it does not turn an acceptance probability $< \frac{1}{4}$ into a probability $> \frac{1}{4}$. \square

C1169 To give a concrete example, here is the transition matrix $M^{(\#_{\sqcup}, \#)} \in \mathcal{M}''$ for the erasing
C1170 pair $(\#_{\sqcup}, \#) \doteq (101\ 100, 101)$:²¹

$$\frac{1}{2^{18}} \begin{pmatrix} 3600 & 12000 & 10000 & | & 15840 & 52800 & 44000 & | & 17424 & 58080 & 48400 & | & 0 & 0 \\ 2400 & 11200 & 12000 & | & 10560 & 49280 & 52800 & | & 11616 & 54208 & 58080 & | & 0 & 0 \\ 1600 & 9600 & 14400 & | & 7040 & 42240 & 63360 & | & 7744 & 46464 & 69696 & | & 0 & 0 \\ \hline 3420 & 11400 & 9500 & | & 15624 & 52080 & 43400 & | & 17820 & 59400 & 49500 & | & 0 & 0 \\ 2280 & 10640 & 11400 & | & 10416 & 48608 & 52080 & | & 11880 & 55440 & 59400 & | & 0 & 0 \\ 1520 & 9120 & 13680 & | & 6944 & 41664 & 62496 & | & 7920 & 47520 & 71280 & | & 0 & 0 \\ \hline 3249 & 10830 & 9025 & | & 15390 & 51300 & 42750 & | & 18225 & 60750 & 50625 & | & 0 & 0 \\ 2166 & 10108 & 10830 & | & 10260 & 47880 & 51300 & | & 12150 & 56700 & 60750 & | & 0 & 0 \\ 1444 & 8664 & 12996 & | & 6840 & 41040 & 61560 & | & 8100 & 48600 & 72900 & | & 0 & 0 \\ \hline 0 & 0 & 0 & | & 0 & 0 & 0 & | & 0 & 0 & 0 & | & \frac{1}{2^{26}} & 2^{18} - \frac{1}{2^{26}} \\ 0 & 0 & 0 & | & 0 & 0 & 0 & | & 0 & 0 & 0 & | & 0 & 2^{18} \end{pmatrix}$$

C1171 7.4 Uniqueness of the solution

C1172 In both parts of Theorem 4, we can achieve that every problem instance that we con-
C1173 struct has a unique solution if it has a solution at all. This comes at the cost of increasing
C1174 the number of matrices and relaxing the bound on the denominators. The Turing ma-
C1175 chine itself is deterministic. The MPCP loses the determinism through the padding pair
C1176 $(\#, \sqcup \# \sqcup)$. We omit this pair and replace it by other word pairs. In particular, if a state
C1177 symbol q is adjacent to the separation symbol $\#$ and is in danger of “falling off” the tape,
C1178 this must be treated as if a \sqcup were present. This leads to one extra word pair for each
C1179 state plus one extra word pair for each left-moving rule.²²

C1180 Since, in addition to the starting pair, also the finishing pair (v_2, w_2) is fixed in the
C1181 2MPCP, the solution to the 2MPCP, and hence the matrix product $M_1 \dots M_m$, becomes

C1182 ²¹If the words v_i and w_i weren't reversed between the MPCP and the RMPCP, the upper left 9×9
C1183 block would be the Kronecker product of the two middle 3×3 blocks in the corresponding 12×12 matrix
C1184 $\bar{M}^{(\#_{\sqcup}, \#)} \in \mathcal{M}''''$ of Proposition 5 for this pair, which was shown on p. 28 in Section 6.5. If we substitute
this Kronecker product as it stands, we get the matrix $\bar{M}^{(\sqcup \#, \#)} \in \mathcal{M}''$ of the opposite erasing pair.

C1185 ²²In contrast to the construction found in most textbooks, we cannot assume that the Turing machine
never moves to the left of its initial position, since we want to keep our Turing machine small.

C1186 unique. (In the normal PCP or MPCP, a solution could be extended by appending
C1187 arbitrary copying pairs.)

C1188 We emphasize that this uniqueness property holds only for output vectors f that are
C1189 constructed according to the proof of Theorem 4. It is obviously impossible to achieve
C1190 uniqueness for every vector $f \in [0, 1]^d$.

C1191 One can check that uniqueness carries over, with the same provisos, to the other
C1192 theorems of this section.

C1193 7.5 Eliminating the output vector, proof of Theorem 2

C1194 We will transfer these results to the classic setting with a *set* of accepting states instead of
C1195 an output vector f . The set of accepting states will be fixed, and the input should come
C1196 through the starting distribution π . Consequently, the Turing machine input should be
C1197 coded, via the first matrix in the matrix product, into the starting distribution π . Hence
C1198 we *reverse* the PCP again, as in Sections 5.7 and 6. We construct a set $\overline{\mathcal{M}}$ of 53 positive
C1199 9×9 matrices, including a matrix for the finishing pair $(H\#\#, \#)$, in the same way as
C1200 in the proof of Theorem 4a, but with reversed words. We refrain from formulating the
C1201 *Doubly-Modified Reversed Post Correspondence Problem* (2MRPCP). We just observe
C1202 that, in the expression for the acceptance probability

$$C1203 \quad e_1^T M^2 M^{a_{m-1}} \dots M^{a_2} M^1 \hat{f} \quad (14)$$

C1204 from (11), the matrix that depends on the input tape u of the Turing machine now
C1205 appears as the matrix M^2 at the beginning of the product, and the matrix that comes
C1206 from the finishing pair $(H\#\#, \#)$ is the last matrix M^1 . Then, $\pi^T := e_1^T M^2$ is the variable
C1207 input to the problem, and $f := M^1 \hat{f}$ is some fixed vector of output values $f_q \in [\frac{1}{4}, \frac{5}{8}]$.
C1208 The acceptance probability becomes

$$C1209 \quad \pi^T M^{a_{m-1}} \dots M^{a_2} f.$$

C1210 What remains to be done is to get rid of the fractional values in the output vector f . We
C1211 will use two methods to convert a PFA with an output vector f with entries from $[0, 1]$ to
C1212 into one with a 0-1 vector f . The first method is a general method that does not change
C1213 the recognized language. It doubles the number of states, and it maintains positivity.²³
C1214 This is formulated as part (a) in the following theorem. As an alternative, we will start
C1215 with the construction of Theorem 4b and we will take the liberty to change the recognized
C1216 language by adding a symbol to the end of every word. This works without adding extra
C1217 states beyond the states q_A and q_R that are already there, and it will lead to part (b) of
C1218 the following theorem.

C1219 **Theorem 2.**

C1220 (a) *There is a fixed set \mathcal{M}' of 52 stochastic matrices of size 18×18 with positive entries*
C1221 *that are multiples of $1/2^{47}$, and a fixed vector $f \in \{0, 1\}^{18}$, for which the following*
C1222 *question is undecidable:*

C1223 *Given a probability distribution $\pi \in \mathbb{Q}^{18}$ whose entries are positive binary fractions,*
C1224 *is there a product $M_1 M_2 \dots M_m$, with $M_j \in \mathcal{M}'$ for all $j = 1, \dots, m$, with*

$$C1225 \quad \pi^T M_1 M_2 \dots M_m f \geq \frac{1}{2} ?$$

C1226 ²³There is a method in the literature with the same effect, but it *squares* the number of states, see [40, proof of Theorem 1, p. 308], [7, Step V, pp. 123–124], or Section 11.2.

C1227 (b) *There is a fixed set \mathcal{M} of 53 stochastic matrices of size 11×11 , all of whose entries*
 C1228 *are multiples of $1/2^{47}$, for which the following question is undecidable:*

C1229 *Given a probability distribution $\pi \in \mathbb{Q}^{11}$ whose entries are binary fractions, is there*
 C1230 *a product $M_1 M_2 \dots M_m$, with $M_j \in \mathcal{M}$ for all $j = 1, \dots, m$, such that*

$$\text{C1231} \quad \pi^T M_1 M_2 \dots M_m e_1 > \frac{1}{4} ?$$

C1232 *In other words, is the language recognized by the PFA with starting distribution π*
 C1233 *and cutpoint $\lambda = \frac{1}{4}$ nonempty?*

C1234 *Proof.* (a) We interpret the output values f_q as probabilities. If we arrive in state q after
 C1235 reading the input, we still have to make a random decision whether to accept the input.
 C1236 The idea is to generate the randomness for making this acceptance decision already *when*
 C1237 each symbol is read, and not *afterwards*, in the end. Every state q of the original PFA
 C1238 comes now in two versions, q^+ and q^- . The transition probabilities to q^+ are multiplied
 C1239 by f_q , and the transition probabilities to q^- are multiplied by $1 - f_q$. The accepting
 C1240 states are the states q^+ .

C1241 In terms of matrices, this can be expressed as follows. Let M be written in column
 C1242 form as

$$\text{C1243} \quad M = (m_1 \ m_2 \ \dots \ m_9).$$

C1244 This is converted to the following 18×18 matrix for the set \mathcal{M}' , arranging the states in
 C1245 the order $q_1^+, \dots, q_9^+, q_1^-, \dots, q_9^-$:

$$\text{C1246} \quad \begin{pmatrix} f_1 m_1 & f_2 m_2 & \dots & f_9 m_9 & (1-f_1)m_1 & (1-f_2)m_2 & \dots & (1-f_9)m_9 \\ f_1 m_1 & f_2 m_2 & \dots & f_9 m_9 & (1-f_1)m_1 & (1-f_2)m_2 & \dots & (1-f_9)m_9 \end{pmatrix}$$

C1247 Similarly, the starting distribution $\pi^T = (\pi_1, \pi_2, \dots, \pi_9)$ is replaced by $(f_1 \pi_1, f_2 \pi_2, \dots,$
 C1248 $f_9 \pi_9, (1-f_1) \pi_1, (1-f_2) \pi_2, \dots, (1-f_9) \pi_9)$. The matrix consists of two equal 9×18 blocks,
 C1249 in accordance with the fact that the distinction between q^+ and q^- has no influence on
 C1250 the next transition.

C1251 As the output values $f_q \in \{\frac{1}{4}, \frac{1}{2}, \frac{5}{8}\}$ are multiples of $\frac{1}{8}$, all resulting probabilities are
 C1252 multiples of $\frac{1}{2^{47}}$.

C1253 (b) The idea is to add to the set of matrices a matrix M^∞ that is necessarily the last
 C1254 matrix in any solution, without imposing this as a constraint.

C1255 We start by constructing a set $\overline{\mathcal{M}''}$ of 53 matrices of size 11×11 , including a matrix
 C1256 for the finishing pair $(H\#\#, \#)$, in the same way as in the proof of Theorem 4b, but with
 C1257 reversed words. The states q_A and q_R are now already present.

C1258 We want to emulate the acceptance criterion of Theorem 4b:

$$\text{C1259} \quad \pi^T M_1 M_2 \dots M_m \hat{f} > \frac{1}{4} \tag{15}$$

C1260 Here, the variable vector π^T as given by (12) has already swallowed the matrix M^2
 C1261 representing the input tape of the Turing machine; However, \hat{f} is the fixed output vector
 C1262 constructed in the proof of Theorem 4b with the values (10) extended by the values
 C1263 $\hat{f}_{q_A} = 1/8$ and $\hat{f}_{q_R} = 0$ for the two additional states. We do not yet merge the last
 C1264 matrix M_m with \hat{f} .

C1265 To the 53 matrices $\overline{\mathcal{M}''}$, we add an extra “final” transition matrix M^∞ . We declare
 C1266 q_A to be the unique accepting state. In the transition M^∞ , each state q goes to q_A with
 C1267 probability \hat{f}_q , and to q_R with the complementary probability $1 - \hat{f}_q$. This rule applies
 C1268 equally to the state q_A , which goes to itself with probability $\hat{f}_{q_A} = 1/8$, and otherwise
 C1269 goes to q_R . The state q_R remains an absorbing state.

C1270 It is clear that adding M^∞ at the end of the product (15) and accepting in state q_A
 C1271 has the same effect as accepting with the output vector \hat{f} . However, a priori we are not
 C1272 sure that M^∞ really comes at the end of the product.

C1273 The acceptance probability of our PFA is given as

$$C1274 \quad \pi^T M_1 M_2 \dots M_{m-1} M_m e_{q_A}, \quad (16)$$

C1275 where the vector of output values is the unit vector e_{q_A} corresponding to the accepting
 C1276 state q_A .

C1277 We will now argue that in any product of this form with matrices M_j from $\overline{\mathcal{M}'''} \cup$
 C1278 $\{M^\infty\}$ that is larger than $\frac{1}{4}$, the matrix M^∞ must appear in the last position M_m , and
 C1279 it cannot appear anywhere else.

C1280 If we never use the matrix M^∞ in the matrix product, the only chance of reaching q_A
 C1281 comes from starting in q_A at the beginning and staying there, and the probability for this
 C1282 is negligibly small. (Even the empty matrix product is not a solution: Remember that
 C1283 π^T is defined in (12) as $\pi^T = \pi_0^T M^2$, where M^2 comes from the word pair representing
 C1284 the input of the Turing machine. Already in M^2 , the probability γ_1 of remaining in q_A ,
 C1285 as given by (13), is very small.)

C1286 On the other hand, when we use the matrix M^∞ , the PFA will arrive in state q_A
 C1287 or q_R . Any further matrices after M^∞ reduce the probability of staying in q_A by a factor
 C1288 $1/8$ or smaller, hence they will not lead to solutions.

C1289 Thus we can assume without loss of generality that M^∞ is the last matrix M_m
 C1290 in the product, and that it is used only in that position. The acceptance probability is
 C1291 then the same as if the output vector \hat{f} had been used instead of M^∞ . (Algebraically,
 C1292 $M^\infty e_{q_A} = \hat{f}$.)

C1293 Thus, the expression (16) has the same value as (15), and it is already of the correct
 C1294 form for our claim. The vector e_{q_A} describes a unique accepting state. As mentioned, we
 C1295 have changed the language recognized by the PFA by adding the symbol corresponding
 C1296 to M^∞ to the end of each word, but this does not affect the emptiness question.

C1297 We can save one matrix by remembering that the last word pair in the PCP solution is
 C1298 always the finishing pair $(v_2, w_2) = (H\#\#, \#)$, and this is used nowhere else. We therefore
 C1299 impose without loss of generality that the corresponding matrix $M^2 = \overline{M}^{(H\#\#, \#)}$ is the
 C1300 penultimate matrix M_{m-1} in the product before M^∞ , and this matrix is used nowhere
 C1301 else. Accordingly, we replace M^2 and M^∞ by one matrix $M^{\text{new}} = M^2 M^\infty$, reducing the
 C1302 number of matrices back to 53. Since the entries of M^∞ are multiples of $\frac{1}{8} = \frac{1}{2^3}$, the
 C1303 entries of the new matrix are multiples of $\frac{1}{2^{27}}$.

C1304 This modification also ensures that the solution is unique: Since we now have enforced
 C1305 that the matrix product (15) ends with $M^2 M^\infty$, in term of the original set of matrices,
 C1306 we are only considering solutions of the MPCP that end with $(H\#\#, \#)$, and these are
 C1307 unique. \square

C1308 7.6 Reduction to 2 input symbols, proof of Theorem 3

C1309 We have already used the reduction to a binary alphabet in the proof of Theorem 1
 C1310 (Section 4.4), but now we will look at an explicit construction. This method has been
 C1311 described, in a more general context, by Hirvensalo in 2007, [17, Step 3 of Section 3] or
 C1312 [16, p. 5], see also Section 11.4.1.

C1313 **Lemma 3.** Consider a PFA A with input alphabet Σ of size $k = |\Sigma| > 2$, and let
 C1314 $\tau: \Sigma^* \rightarrow \{\mathbf{a}, \mathbf{b}\}^*$ be a coding function using the codewords $\mathbf{b}, \mathbf{ab}, \mathbf{aab}, \dots, \mathbf{a}^{k-2}\mathbf{b}, \mathbf{a}^{k-1}$.

C1315 Then there is a PFA A' with input alphabet $\{\mathbf{a}, \mathbf{b}\}$ that accepts each word $\tau(u)$ with the
 C1316 same probability as A accepts $u \in \Sigma^*$. Words that are not of the form $\tau(u)$ are accepted
 C1317 with probability 0.

C1318 *The number of states is multiplied by $k - 1$ in this construction.*

C1319 *Proof.* Suppose A has transition matrices M_1, \dots, M_k corresponding to the k input sym-
 C1320 bols. We construct a PFA A' that does the decoding in a straightforward way. It
 C1321 maintains the number of \mathbf{a} 's that have been seen in a counter variable i in the range
 C1322 $0 \leq i \leq k - 2$. In addition, it maintains the state $q \in Q$ of the original PFA A . Thus, the
 C1323 state set of A' is $Q' = \{0, \dots, k - 2\} \times Q$. Initially, q is chosen according to the starting
 C1324 distribution of A , and $i = 0$.

C1325 • If A' reads the letter \mathbf{b} , it changes the state q to a random new state according to
 C1326 the transition matrix M_{i+1} , and resets $i := 0$.

C1327 • If A' reads an \mathbf{a} and $i < k - 2$, it increments the counter: $i := i + 1$.

C1328 • If A' reads an \mathbf{a} and $i = k - 2$, it changes the state q to a random new state
 C1329 according to M_k , and resets $i := 0$.

C1330 An input is accepted if $i = 0$ and q is an accepting state of A .

C1331 The transition matrices for the symbols \mathbf{a} and \mathbf{b} can be written in block form as

$$M'_a = \begin{pmatrix} 0 & I & & & \\ & 0 & I & & \\ & & 0 & \ddots & \\ & & & \ddots & 0 & I \\ M_k & & & & & 0 \end{pmatrix} \text{ and } M'_b = \begin{pmatrix} M_1 & 0 & \cdots & 0 \\ M_2 & 0 & \cdots & 0 \\ M_3 & 0 & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ M_{k-1} & 0 & \cdots & 0 \end{pmatrix}. \quad \square$$

C1332 The construction works more generally for any prefix-free code. The set of states Q'
 C1333 will have the form $K \times Q$, where the states in K do the decoding.

C1334 Applying this to Theorem 2b, we get:

C1335 **Theorem 3.** *There is a PFA with 572 states, two input symbols with fixed transition*
 C1336 *matrices, all of whose entries are multiples of $1/2^{47}$, and with a single accepting state,*
 C1337 *for which the following question is undecidable:*

C1338 *Given a probability distribution $\pi \in \mathbb{Q}^{572}$ whose entries are binary fractions, is the*
 C1339 *language recognized by the PFA with starting distribution π and cutpoint $\lambda = \frac{1}{4}$ nonempty?*

C1340 The number $572 = 52 \times 11$ of states is an overcount. For example, the absorbing
 C1341 state q_R can be left as is and need not be multiplied with 52.

C1342 If we are more ambitious, we can achieve that all matrix entries are from the set
 C1343 $\{0, \frac{1}{2}, 1\}$, as in Theorem 1, instead of multiples of 2^{-47} . We apply the technique from
 C1344 item (b) in the proof of that theorem (Section 4.4): We simply add a block of 46 padding
 C1345 \mathbf{a} 's after every codeword.²⁴

C1346 ²⁴We can roughly estimate the required number of states as follows. Let $|Q| = 11$ be the number of
 C1347 states of the original automaton, and let $k = |\Sigma| = 53$ be its number of symbols. For each combination in
 C1348 $Q \times \Sigma$, whenever the algorithm in Lemma 3 asks to “change the state q to a random new state according
 C1349 to M_i ”, we have to set up a binary decision tree of height 47 to determine the next state. We can think
 C1350 of this tree as follows: For a random number $x = 0.x_1x_2 \dots x_{47}$, we want to determine which of $|Q|$
 C1351 intervals $[0, c_1], (c_1, c_2], (c_2, c_3], \dots, (c_{|Q|-1}, 1]$ contains x , by looking at the successive bits x_j of x . This
 C1352 tree has at most $(|Q| - 1) \times 46$ nodes where the outcome has not been decided: each such node lies
 C1353 on a root-to-leaf path to some interval endpoint c_i . In addition we need up to $46 \times |Q|$ states for the
 C1354 situation when the next state has been decided and the algorithm only needs to count to the end of the
 C1355 padding block. In total, this gives an upper bound of $(k - 1)|Q| + |Q|k(|Q| - 1)46 + 46|Q|$ states, which
 C1356 is $572 + 46 \times 11 \times (53 \times 10 + 1) = 269\,258$.

C1357

8 Using integer matrices

C1358

C1359

C1360

C1361

As an alternative to constructions involving PFAs only, we start from matrix product problems of arbitrary *integer* matrices. These can be converted to stochastic matrices at some cost, but overall, we get the automata with the smallest number of states that are known for having an undecidable Emptiness Problem.

C1362

8.1 The smallest number of states without regard to the size of the alphabet, Theorem 5

C1363

C1364

C1365

Theorem 5 (Claus [8, Theorem 6(iii), p. 151]). *The PFA Emptiness Problem (1) is undecidable for PFAs with a deterministic start state, a single accepting state, and 5 positive transition matrices of size 9×9 , and with cutpoint $\lambda = 1/9$.*

C1366

C1367

C1368

Proof. Neary [24] has shown that the PCP with five word pairs is undecidable. Suppose the word pairs $(v_1, w_1), (v_2, w_2), (v_3, w_3), (v_4, w_4), (v_5, w_5)$ are encoded over the alphabet $\Sigma = \{1, 2\}$. We denote the base-3 value of $u = u_1 u_2 \dots u_n \in \Sigma^*$ by $(u)_3 = \sum_{j=1}^n u_j 3^{n-j}$.

C1369

8.1.1 Step 5.1. Modeling the PCP by integer matrices

C1370

C1371

Following Claus [8, Definition 5, p. 143], we define the following 6×6 matrix $A(v, w)$ for words $v, w \in \{1, 2\}^*$.

$$A(v, w) = \begin{pmatrix} 1 & -2(v)_3 & 2(w)_3 & ((v)_3)^2 & ((w)_3)^2 & -2(v)_3(w)_3 \\ 0 & 3^{|v|} & 0 & -(v)_3 \cdot 3^{|v|} & 0 & (w)_3 \cdot 3^{|v|} \\ 0 & 0 & 3^{|w|} & 0 & (w)_3 \cdot 3^{|w|} & -(v)_3 \cdot 3^{|w|} \\ 0 & 0 & 0 & 3^{2|v|} & 0 & 0 \\ 0 & 0 & 0 & 0 & 3^{2|w|} & 0 \\ 0 & 0 & 0 & 0 & 0 & 3^{|v|+|w|} \end{pmatrix} \quad (17)$$

C1372

C1373

It is not straightforward to see, but can be checked by a calculation that A satisfies the multiplicative law [8, Lemma 2]

C1374

$$A(v_1, w_1)A(v_2, w_2) = A(v_1 v_2, w_1 w_2). \quad (18)$$

C1375

With the vectors $e_1^T = (1, 0, 0, 0, 0, 0)$ and $f_1^T = (1, 0, 0, -1, -1, -1)$, one gets

C1376

$$e_1^T A(v, w) f_1 = 1 - ((v)_3 - (w)_3)^2. \quad (19)$$

C1377

C1378

This expression checks whether $v = w$: Since all data are integral, $e_1 A(v, w) f_1 > 0$ iff $v = w$.

C1379

C1380

C1381

From the PCP word pairs (v_i, w_i) , we now construct the matrices $B_i = A(v_i, w_i)$. By repeated application of (18), $A(v_{a_1} \dots v_{a_m}, w_{a_1} \dots w_{a_m}) = B_{a_1} B_{a_2} \dots B_{a_m}$. Thus, the index sequence $a = a_1 a_2 \dots a_m$ is a solution of the PCP if and only if

C1382

$$e_1^T B_{a_1} B_{a_2} \dots B_{a_{m-1}} B_{a_m} f_1 > 0. \quad (20)$$

C1383

8.1.2 Step 5.2. Introducing a unit vector as an ending vector

C1384

C1385

C1386

C1387

This step and the next one are a standard part of the proof of Turakainen's Theorem [40, 41] (mentioned in the introduction of Section 7), and, in one form or another, they are described in many places, see [7, Section 3.3.3, p. 120–125], [3, Steps 4b–4c in Section 2, p. 237–238], or [17, Steps 4–6 in Section 3].)

C1388 First we essentially introduce a new final state (state 7), extending the matrices to
 C1389 size 7×7 :

$$C1390 \quad D_i = \begin{pmatrix} B_i & B_i f_1 \\ 0 & 0 \end{pmatrix}. \quad (21)$$

C1391 One can then check that

$$C1392 \quad D_{a_1} \dots D_{a_m} = \begin{pmatrix} B_{a_1} \dots B_{a_m} & B_{a_1} \dots B_{a_m} f_1 \\ 0 & 0 \end{pmatrix}. \quad (22)$$

C1393 With the final vector $f_2 = e_7$, the expression

$$C1394 \quad e_1^T D_{a_1} D_{a_2} \dots D_{a_{m-1}} D_{a_m} e_7, \quad (23)$$

C1395 which represents the upper right corner of the matrix product (22), has the same value
 C1396 as the expression in (20) whenever we have a nonempty matrix product.²⁵ For $m = 0$,
 C1397 (23) has value 0, and thus we have now excluded the empty PCP solution.

C1398 8.1.3 Step 5.3. Conversion to stochastic matrices

C1399 The conversion of the matrices to stochastic matrices will introduce two more states.

C1400 As a preparation, we extend the matrices D_i by two extra rows and columns to the
 C1401 matrices

$$C1402 \quad E_i = \begin{pmatrix} D_i & 0 & t_i \\ r_i^T & 0 & s_i \\ 0 & 0 & 0 \end{pmatrix},$$

C1403 where the vectors r_i and t_i and the scalar s_i are chosen to make all row and column sums
 C1404 zero. It can be checked that the product $E_i E_j$ of two such matrices has the same form,
 C1405 its row and column sums are zero, and its upper left corner contains the matrix $D_i D_j$.
 C1406 We extend the starting vector e_1 and the output vector e_7 by adding two zeros at the
 C1407 end. Due to this setup, the extra rows and columns play no role for the result. We have
 C1408 thus transformed (20) into the equivalent condition

$$C1409 \quad e_1^T E_{a_1} E_{a_2} \dots E_{a_m} e_7 > 0, \quad (24)$$

C1410 The matrices E_i have dimension $d = 9$. Let J denote the doubly-stochastic $d \times d$ matrix
 C1411 of the “completely random” transition with all entries equal to $1/d$. Then, with a small
 C1412 constant $\alpha > 0$, we form the matrices $F_i := J + \alpha E_i$. The constant α is chosen small
 C1413 enough such that $F_i > 0$ for all matrices F_i . Since the row sums are 1, the matrices F_i
 C1414 are now stochastic.

C1415 The product of the new matrices F_i is

$$C1416 \quad e_1^T F_{a_1} F_{a_2} \dots F_{a_m} e_7 = e_1^T J e_7 + \alpha^m e_1^T E_{a_1} E_{a_2} \dots E_{a_m} e_7.$$

C1417 The reason is that $E_i J = J E_i = 0$, and hence all “mixed” terms in the expansion of
 C1418 the product $\prod F_{a_i} = \prod (J + \alpha E_{a_i})$ vanish. The “pure” product J^m simplifies due to the
 C1419 relation $J J = J$. Since $e_1^T J e_7 = 1/d = 1/9$, (24) becomes equivalent to

$$C1420 \quad e_1^T F_{a_1} F_{a_2} \dots F_{a_m} e_7 > \frac{1}{9}.$$

C1421 The matrices F_i are positive by construction. This proves the result. \square

C1422 ²⁵Actually, it is the 7×7 matrices (21) that Claus takes as the starting point [8, Definition 5, $\phi(u, v)$,
 C1423 p. 143]. In his proof, there is no analog of Step 5.1 that would be separate from Step 5.2. He investigated
 C1424 the “POGAMOR-problem” of checking whether the upper right corner of a product of matrices can be
 C1425 made positive.

C1426

8.2 Binary alphabet, Theorem 6

C1427

C1428

C1429

For reducing the size of the input alphabet (or the number of matrices) to the absolute minimum, namely two, we modify some steps of the previous proof and combine them with other steps.

C1430

C1431

C1432

C1433

Theorem 6 ([17]). *The PFA Emptiness Problem (1) is undecidable for PFAs with two positive transition matrices size 20×20 , a single deterministic start state, a single accepting state, and with cutpoint $\lambda = 1/20$.*

The same is true with weak inequality ($\geq 1/20$) as the acceptance criterion.

C1434

Proof. We closely follow the proof of Hirvensalo [17].

C1435

8.2.1 Step 6.1. Modeling the PCP by integer matrices

C1436

C1437

We use the following variation of the matrix A from (17). (It has been reflected at the SW–NE diagonal.)

$$\tilde{A}(v, w) = \begin{pmatrix} 3^{|v|+|w|} & 0 & 0 & -(v)_3 \cdot 3^{|w|} & (w)_3 \cdot 3^{|v|} & -2(v)_3(w)_3 \\ 0 & 3^{2|w|} & 0 & (w)_3 \cdot 3^{|w|} & 0 & ((w)_3)^2 \\ 0 & 0 & 3^{2|v|} & 0 & -(v)_3 \cdot 3^{|v|} & ((v)_3)^2 \\ 0 & 0 & 0 & 3^{|w|} & 0 & 2(w)_3 \\ 0 & 0 & 0 & 0 & 3^{|v|} & -2(v)_3 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

C1438

It also satisfies the multiplicative law, but with reversed order of factors:

C1439

$$\tilde{A}(v_1, w_1)\tilde{A}(v_2, w_2) = \tilde{A}(v_2v_1, w_2w_1)$$

C1440

C1441

C1442

C1443

C1444

Thus it works in the same way as the matrix A .²⁶ The important feature is that the last row is a unit row, and thus it acts like an absorbing state. This will allow us to save two states when reducing the number of matrices to two.

We can test equality of strings $v = w$ in terms of the matrix \tilde{A} with a starting vector $\pi_1 = (-2, -2, -2, 0, 0, 1)^T$ and an output vector $f_1 = (0, 0, 0, 0, 0, 1)^T$:

C1445

$$\pi_1^T \tilde{A}(v, w) f_1 = 1 - 2((v)_3 - (w)_3)^2$$

C1446

C1447

Thus, since all data are integral, $\pi_1 \tilde{A}(v, w) f_1 > 0$ iff $v = w$. The value 0 cannot appear, and thus the construction works equally with the weak inequality $\pi_1 \tilde{A}(v, w) f_1 \geq 0$.²⁹

C1448

C1449

C1450

C1451

C1452

C1453

C1454

C1455

C1456

C1457

C1458

C1459

C1460

C1461

C1462

C1463

C1464

²⁶Our definition of the matrix \tilde{A} differs from the corresponding matrix $A(u, v)$ in [3, p. 235]) and the matrix $\gamma(u, v)$ in [17, Eq. (4)] in some nonessential details. Most importantly, the matrix is transposed. In addition, the first and second rows, and the first and second columns are swapped. The matrices in [3] and [17] have no negative signs. Blondel and Canterini [3] use radix 10 instead of radix 3,²⁷ and Hirvensalo [17] uses radix 2.²⁸ We have chosen the form of \tilde{A} to remain consistent with [8] and (17).

In the matrix formulation of the acceptance probability of a PFA in Hirvensalo [17, Eq. (1), p. 310], the starting distribution π (or \mathbf{y} , in the notation of [17]) is written on the right end of the matrix product and the output vector f (or \mathbf{x}) on the left, and the transition matrices are column-stochastic. Thus, despite the fact that the matrices are transposed, our interpretation as a PFA is the same as Hirvensalo's.

²⁷"The notation is quaternary, decimal, etc., according to taste." (Paterson [29])

²⁸The reader may worry that the binary value function $(u)_2$ might not be injective for strings u with ternary digits 1, 2. In fact, the function $(u)_2$ is even a *bijection* between the strings $u \in \{1, 2\}^*$ and the nonnegative integers. This can be extended to any radix.

²⁹Hirvensalo uses the starting vector $(-1, -1, -1, 0, 0, 1)$ (translated to our formulation), and this works only for acceptance with strict inequality. He treats the weak acceptance variant (with $\geq \lambda$) by adjusting the matrix later, which requires an additional state [17, Modification Step 3.5, p. 315]. Alternatively, he could have used the starting vector $(-1, -1, -1, 0, 0, 0)$.

C1465 For a PCP with word pairs (v_i, w_i) , we thus form the matrices $B_i := \tilde{A}(v_i, w_i)$, and
 C1466 then the PCP solutions $a_1 a_2 \dots a_m$ (including the empty solution) are characterized by
 C1467 the inequality

$$C1468 \quad \pi_1^T B_{a_m} B_{a_{m-1}} \dots B_{a_2} B_{a_1} f_1 > 0.$$

C1469 8.2.2 Step 6.2. Merging the first and last matrices into the boundary vectors

C1470 Hirvensalo [17] based his proof on the undecidable PCP instances with 7 word pairs of
 C1471 Matiyasevich and Sénizergues [20], which had the smallest number of pairs until 2015.
 C1472 These instances are actually instances of what we have called the Doubly-Modified Post
 C1473 Correspondence Problem (2MPCP, Section 7.3). They have two special pairs (v_1, w_1) and
 C1474 (v_2, w_2) , which must be used at the beginning and at the end of the solution, respectively,
 C1475 and can be used nowhere else. Thus, as in (11), the two corresponding matrices B_1 and
 C1476 B_2 can be multiplied with π_1 and f_1 , respectively, yielding new matrices $\pi_2^T = \pi_1^T B_2$ and
 C1477 $f_2 := B_1 f_1$. This reduces the number of matrices, and thus the size of the alphabet, from
 C1478 7 to 5. By this change, we have also eliminated the unwanted empty solution ($m = 0$).

C1479 We can reduce the number of matrices by substituting the undecidable PCP instance
 C1480 of Neary [24] with only 5 pairs. In these instances, there is also a starting pair (v_1, w_1)
 C1481 and an ending pair (v_2, w_2) , such that every solution necessarily starts with (v_1, w_1) and
 C1482 ends with (v_2, w_2) . The ending pair can appear nowhere else. However, unlike in the
 C1483 2MPCP, the starting pair is also used in the middle of the solutions. (This multipurpose
 C1484 usage of the starting pair is one of the devices to achieve such a remarkably small number
 C1485 of word pairs.³⁰) Thus, we can merge the boundary matrices into π_1 and f_1 as above.
 C1486 But this reduces the number of matrices only from 5 to 4.

C1487 Thus, we are left with four 6×6 integer matrices, which we call C_1, C_2, C_3, C_4 .

C1488 8.2.3 Step 6.3. Reduction to two matrices

C1489 We have in integer-weighted automaton with 6 states and inputs $a_{m-1} a_{m-2} \dots a_2 \in$
 C1490 $\{1, 2, 3, 4\}^*$ from an alphabet of 4 symbols. We reduce the input alphabet to a binary
 C1491 alphabet along the lines of Lemma 3, using the codewords **b**, **ab**, **aab**, **aaa**.

C1492 Applying Lemma 3 directly would multiply the number of states by 3. However, we
 C1493 observe that state 6 behaves like an absorbing state, because the last line of all matrices
 C1494 is the unit vector that leads back to state 6. Thus, when the automaton is in state 6, we
 C1495 can stop decoding the input, and we need not split state 6 into three states.

C1496 Formally, we split the 6×6 matrices C_i and the vector π_2 and f_2 into blocks of size
 C1497 $5 + 1$:

$$C1498 \quad C_i = \begin{pmatrix} \hat{C}_i & c_i \\ 0 & 1 \end{pmatrix}, \pi_2 = \begin{pmatrix} \hat{\pi}_2 \\ p_2 \end{pmatrix}, f_2 = \begin{pmatrix} \hat{f}_2 \\ g_2 \end{pmatrix}.$$

C1499 Following the construction in the proof of Lemma 3, the new transition matrices and

C1500 ³⁰The proof of Theorem 11 in Neary [24] contains a mistake which needs to be fixed: Neary's PCP
 C1501 instances encode binary tag systems. When showing that the PCP solution must follow the intended
 C1502 patterns of the simulation of the binary tag system, Neary [24, p. 660] needs to show that the ending pair
 C1503 $(v_5, w_5) = (10^\beta 1111, 1111)$ cannot be used except to bring the two strings to a common end. He claims
 C1504 that a block 1111 cannot appear in the encoded string because in u (the unencoded string of the binary
 C1505 tag system, which is described in Lemma 9) we cannot have two c symbols next to each other. This is
 C1506 not true. The paper contains plenty of examples, and they contradict this claim; for example, the string
 C1507 u' in (7) [24, p. 657] contains seven c 's in a row. The mistake can be fixed by taking a longer block of 1s:
 C1508 Looking at the appendants in Lemma 9, it is clear that every block of length $|u| + 1$ must contain a b .
 C1509 Thus the word pair $(v_5, w_5) = (10^\beta 1^{|u|+99}, 1^{|u|+99})$ will work.

C1510 starting and ending vectors are written in block form with block sizes $5 + 5 + 5 + 1 = 16$:

$$C1511 \quad M'_a = \begin{pmatrix} 0 & I & 0 & 0 \\ 0 & 0 & I & 0 \\ \hat{C}_4 & 0 & 0 & c_4 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \quad M'_b = \begin{pmatrix} \hat{C}_1 & 0 & 0 & c_1 \\ \hat{C}_2 & 0 & 0 & c_2 \\ \hat{C}_3 & 0 & 0 & c_3 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \quad \pi_3 = \begin{pmatrix} \hat{\pi}_2 \\ 0 \\ 0 \\ p_2 \end{pmatrix}, \quad \text{and } f_3 = \begin{pmatrix} \hat{f}_2 \\ \hat{f}_2 \\ \hat{f}_2 \\ g_2 \end{pmatrix}.$$

C1512 With the intermediate product

$$C1513 \quad M'_a M'_a = \begin{pmatrix} 0 & 0 & I & 0 \\ \hat{C}_4 & 0 & 0 & c_4 \\ 0 & \hat{C}_4 & 0 & c_4 \\ 0 & 0 & 0 & 1 \end{pmatrix},$$

C1514 one can work out the matrices M'_b , $M'_a M'_b$, $M'_a M'_a M'_b$, and $M'_a M'_a M'_a$ and check that they
C1515 are of the form

$$C1516 \quad \begin{pmatrix} \hat{C}_i & 0 & 0 & c_i \\ * & * & * & * \\ * & * & * & * \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

C1517 for $i = 1, 2, 3, 4$, thus simulating the original automaton on the states 1–5 and 16: It is
C1518 easy to establish by induction that multiplying the initial distribution π_3^T with a sequence
C1519 of such matrices produces a vector x^T of the form

$$C1520 \quad x^T = (\hat{x}^T \ 0 \ 0 \ x_6) \tag{25}$$

C1521 whose nonzero entries $\underline{x}^T = (\hat{x}^T \ x_6)$ coincide with the entries of the corresponding vector
C1522 produced from π_2^T with the original matrices C_i . If the scalar product with f_3 is taken,
C1523 the result $\hat{x}^T \hat{f}_2 + x_6 g_2$ is the same as with the original vectors \underline{x}^T and f_2 .

C1524 One technicality remains to be discussed: Some “unfinished” words in $\{\mathbf{a}, \mathbf{b}\}^*$ do not
C1525 factor into codewords but end in a partial codeword \mathbf{a} or \mathbf{aa} . To analyze the corresponding
C1526 matrix products, we also look at the products M'_a , and $M'_a M'_a$. They have the form

$$C1527 \quad \begin{pmatrix} 0 & I & 0 & 0 \\ * & * & * & * \\ * & * & * & * \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad \text{and} \quad \begin{pmatrix} 0 & 0 & I & 0 \\ * & * & * & * \\ * & * & * & * \\ 0 & 0 & 0 & 1 \end{pmatrix},$$

C1528 and therefore, multiplying them with the vector x^T of (25) yields $(0 \ \hat{x}^T \ 0 \ x_6)$ and
C1529 $(0 \ 0 \ \hat{x}^T \ x_6)$, respectively. If this is multiplied with f_3 , the result is still the same as
C1530 with the vector x^T of (25). Thus, *input sequences whose decoding process leaves a partial*
C1531 *codeword \mathbf{a} or \mathbf{aa} at the end produce the same value as if that partial codeword were*
C1532 *omitted*. This is the desired behavior in the context of the emptiness question.³¹ We
C1533 can describe the issue in terms of the algorithm of Lemma 3: The acceptance decision of
C1534 this algorithm takes into account the value of the counter i : It accepts only when $i = 0$.
C1535 However, in state 6, our algorithm “loses track” of i . Thus we have to ignore the counter
C1536 also in the other states. This is reflected in the vector f_3 by copying \hat{f}_2 for every value
C1537 of the counter ($i = 0, 1, 2$).

C1538 ³¹ We are thus committing the (inconsequential) “error” which took Nasu and Honda [23] and Paz
C1539 [30] some effort to fix, see Section 11.4. Hirsensalo [17, p. 314] defined the vector f_3 (\mathbf{y}_3 in his notation)
C1540 “analogously” to the vector π_3 (\mathbf{x}_3 in his notation), thus by filling the middle entries with zeros. Then
C1541 incomplete inputs give a value < 0 . The argument is rather delicate: We see that with this definition, the
C1542 result for incomplete inputs is $x_6 g_2$ (in our notation) instead of $\hat{x}^T \hat{f}_2 + x_6 g_2$. One can check that $g_2 = 1$;
C1543 Hence the result is as if the original vector $f_1 = (0, 0, 0, 0, 0, 1)^T$ had been used instead of $f_2 := B_1 f_1$.
C1544 The result is therefore of the form $1 - 2((v)_3 - (w)_3)^2$, where v and w are composed of a nonempty
C1545 sequence of word pairs (v_i, w_i) *without* the the special start pair (v_1, w_1) . By the properties of the PCP,
C1546 we can never have $v = w$, and thus $x_6 g_2$ is indeed negative.

C1547 **8.2.4 Step 6.4. Introducing unit vectors as starting and ending vectors**

C1548 We have now two 16×16 matrices M'_a , and M'_b and the corresponding starting and
C1549 ending vectors π_3 and f_3 .

C1550 This step and the next one follow the standard proof of Turakainen's Theorem [40, 41],
C1551 and they have already been described in Section 8.1.

C1552 First we introduce a new start state (state 1) and a new final state (state 18), ex-
C1553 tending the matrices to size 18×18 . This is similar to Step 5.2 (Section 8.1.2), where
C1554 we introduced only a final state, because the starting vector was already a unit vector.

C1555 We set $\pi_4 = e_1$, $f_4 = e_{18}$, and

$$C1556 \quad D_1 = \begin{pmatrix} 0 & \pi_3^T M'_a & \pi_3^T M'_a f_3 \\ 0 & M'_a & M'_a f_3 \\ 0 & 0 & 0 \end{pmatrix}, \quad D_2 = \begin{pmatrix} 0 & \pi_3^T M'_b & \pi_3^T M'_b f_3 \\ 0 & M'_b & M'_b f_3 \\ 0 & 0 & 0 \end{pmatrix}.$$

C1557 One can check that this leads to the same results.

C1558 **8.2.5 Step 6.5. Conversion to stochastic matrices**

C1559 This conversion has been described above in Step 5.3 (Section 8.1.3). It introduces two
C1560 more states, leading to a total of 20 states. This concludes the proof of Theorem 6. \square

C1561 **8.3 History of ideas**

C1562 Mike Paterson [29] pioneered the modeling of the PCP by multiplication of integer ma-
C1563 trices in 1970 in order to show that mortality for 3×3 matrices is undecidable. This
C1564 problem asks whether the zero matrix can be obtained as a product of matrices from a
C1565 given set. Paterson introduced the matrix

$$C1566 \quad \begin{pmatrix} 10^{|v|} & 0 & 0 \\ 0 & 10^{|w|} & 0 \\ (v)_{10} & (w)_{10} & 1 \end{pmatrix}, \quad (26)$$

C1567 to represent a pair (v, w) of a PCP.³² These matrices satisfy a multiplicative law like (18).

C1568 It was pointed out both by Claus [8, p. 153] and by Blondel and Canterini [3, p. 235]
C1569 that one can generate the quadratic terms that are necessary to form the expression
C1570 $1 - ((v)_{10} - (w)_{10})^2$ by taking the Kronecker product of two 3×3 matrices of the
C1571 form (26), which would be a 9×9 matrix. The matrix (17) of smaller size 6×6 , which
C1572 is sufficient for this task, was found by Claus [8, p. 153] in 1981, and in slightly different
C1573 form by Blondel and Canterini [3] in 2003, see footnote 26.

C1574 As repeatedly mentioned, the conversion from general integer matrices and vectors π
C1575 and f to the PFA setting is due to Turakainen [40] from 1969. The techniques were later
C1576 sharpened by Turakainen [41] in 1975.

C1577 We mention that Theorem 5 can be slightly strengthened by using techniques from
C1578 the proof of Theorem 6. Neary's PCP is an RMPCP with an ending pair that that
C1579 is used nowhere else. Thus, we can merge the rightmost matrix B_m with the vector
C1580 f_1 , analogous to Step 6.2 (Section 8.2.2). If this is done right at the beginning, *before*
C1581 introducing a new final state (Step 5.2, Section 8.1.2), it does not affect the remainder
C1582 of the proof. This reduces the number of matrices in Theorem 5 from 5 to 4. At the
C1583 same time, this eliminates the empty PCP solution. In addition, with the alternative
C1584 final vector $f_1^T = (0, 0, 0, -1, -1, -1)$ instead of $(1, 0, 0, -1, -1, -1)$ in (19), the theorem
C1585 extends to weak inequality ($\geq 1/9$) as the acceptance criterion.

³²Paterson specified these matrices informally by way of example, without committing to a particular base: "The notation is quaternary, decimal, etc., according to taste."

C1587

9 Alternative universal Turing machines

C1588

C1589

C1590

C1591

Our proofs rely on a particular small universal Turing machine. In the literature, some “universal” Turing machines with smaller numbers of states and symbols are proposed. We review these machines and discuss whether they could possibly be used to decrease the number of matrices in Theorems 2–4.

C1592

9.1 Watanabe, weak and semi-weak universality

C1593

C1594

C1595

C1596

C1597

C1598

C1599

A universal Turing machine U_W with 3 symbols and 7 states was published by Shigeru Watanabe [43] in 1972, but I haven’t been able to get hold of this paper. According to the survey [45, Fig. 1], this is a *semi-weakly* universal Turing machine. In *semi-weakly* and *weakly* universal machines, the empty parts of the tape on one or both sides of the input are initially filled with some repeating pattern instead of uniformly blank symbols. Such a repeating pattern can be easily accommodated in the translation to the MPCP by modifying the padding pair of words $(\#, \sqcup\#\sqcup)$.

C1600

C1601

C1602

C1603

C1604

C1605

In the worst case, the 21 rules contain only one halting rule and the remaining 20 rules are balanced between left- and right-moving rules. Then, with 10 left-moving rules, 1 halting rule, 10 right-moving rules, and $|\Gamma| = 3$, we get $3 \times 3 + 3 + 10 \times 3 + 11 = 53$ matrices, the same number as from the machine $U_{15,2}$ of Neary and Woods. Any imbalance in the distribution of left-moving and right-moving rules would allow to reduce the number of matrices in Theorem 2b from 53 to 51 or less.

C1606

C1607

C1608

C1609

C1610

This speculative improvement depends on an assumption, which would need to be verified. According to [45, Section 3.1], Watanabe’s weak machine U_W simulates other Turing machines T directly. What would be most useful for us is that the periodic pattern that initially fills the tape of U_W is a fixed pattern that is specified as part of the definition of U_W and does not depend on T or its input.

C1611

C1612

C1613

C1614

C1615

C1616

C1617

If this is the case, we can use them for our construction, where only the first (or last) pair of the PCP should depend on the input.

We could even accommodate some weaker requirement, namely that the periodic pattern depends on the Turing machine T that is being simulated, as long as it is independent of the input u to that Turing machine. We could then let U_W simulate a fixed universal Turing machine $T = U_0$ (universal in the usual, standard, sense), and then the periodic pattern would also be fixed.

C1618

9.2 Wolfram–Cook, rule 110

C1619

C1620

C1621

C1622

C1623

C1624

C1625

C1626

C1627

C1628

C1629

C1630

C1631

Some small machines are based on simulating a particular cellular automaton, the so-called *rule-110 automaton* of Stephen Wolfram. These machines are given in [11, Fig. 1, p. 3] and [27], see also the survey [45]. The machines of [27] have as few as 6 states and 2 symbols, or 3 states and 3 symbols, or 2 states and 4 symbols. The rule-110 automaton was shown to be universal by Matthew Cook [11], see also Wolfram [44, Section 11.8, pp. 675–689].³³The universality of the rule-110 automaton comes from the fact that rule 110 can simulate *cyclic tag systems*. *Tag systems* are a special type of string rewriting systems, where symbols are deleted from the front of a string, and other symbols are appended to the end of a string, according to certain rules. *Cyclic tag systems* are a particularly simple variation of tag systems. Tag systems as well as cyclic tag systems are known to be universal, because they can simulate Turing machines.

One difficulty with these small Turing machines, which makes them not directly suitable for our purposes is that, like in the weakly universal machines of Section 9.1,

C1632

³³on-line at <https://www.wolframscience.com/nks/p675--the-rule-110-cellular-automaton/>

C1633 the repeating patterns by which the ends of the tape are filled are not fixed, but depend
 C1634 on the tag system, see [44, Note on initial conditions, p. 1116].³⁴ As a consequence, we
 C1635 don't have a fixed replacement for the padding pair $(\#, \sqcup\#\sqcup)$. Thus we cannot use them
 C1636 for the proof of Theorems 2 and 4, where only the first (or last) pair of the PCP should
 C1637 depend on the input.

C1638 As in Section 9.1, one could start with a universal Turing machine U_0 and construct
 C1639 from it a fixed cyclic tag system. The classic way to do this is the method of Cocke and
 C1640 Minsky from 1964 [9]. It converts a Turing machine U to a tag system \mathcal{C} . The transition
 C1641 rules of U are converted to the rules (appendants) of \mathcal{C} , and the input tape of U is
 C1642 translated to the starting string of the tag system \mathcal{C} . If we start with a universal Turing
 C1643 machine U_0 , the rules of \mathcal{C} are fixed. The simulation of the tag system \mathcal{C} by a cyclic tag
 C1644 system \mathcal{C}' is easy [11].³⁵ Eventually, in the Turing machine T that simulates the rule-111
 C1645 automaton via the cyclic tag system \mathcal{C}' , this translates to a fixed pattern by which the
 C1646 empty tape of T is filled on the two sides of the input. Therefore, the padding pair
 C1647 $(\#, \sqcup\#\sqcup)$ in the PCP has a fixed replacement, which is translated into a fixed stochastic
 C1648 matrix for the PFA.

C1649 There is, however, another reason why the machines simulating the rule-110 automa-
 C1650 ton cannot be used directly: They have no provision for *halting*, or for otherwise deter-
 C1651 mining some set of inputs that they accept.³⁶ This is natural in the context of a cellular
 C1652 automaton, which performs an infinite process. However, a cyclic tag system, which the
 C1653 automaton supposedly simulates, *does* have a way of terminating, namely when the string
 C1654 on which it operates becomes empty. Fortunately, Cook gives a few hints about termina-
 C1655 tion and about undecidable questions for the corresponding Turing machines: *Questions*
 C1656 *about their behavior, such as "Will this sequence of symbols ever appear on the tape?",*
 C1657 *are undecidable* [11, Note [7], p. 38]. More specifically, Cook mentions some particular
 C1658 undecidable questions for so-called *glider systems*. Some consequences of this discussion
 C1659 for Rule 110 are briefly touched upon in [11, Section 4.6, p. 37]: *So another specific*
 C1660 *example of an undecidable question for Rule 110 is: Given an initial middle segment,*
 C1661 *will there ever be an F ?* Here, an F is a particular type of "glider", a cyclically repeated
 C1662 sequence of patterns that moves at constant speed through the cellular automaton as
 C1663 long as it does not hit other gliders or irregularities. Hypothesizing that the presence of
 C1664 such a glider could be detected by the occurrence of a particular pattern \hat{F} in the cellular
 C1665 automaton, or on the Turing machine tape, such a criterion could be translated into a
 C1666 word pair (\hat{F}, H) that introduces the halting symbol H , and this would lead to small
 C1667 undecidable instances of the MPCP.

C1668 All these arguments require careful examination, and the approach depends on a
 C1669 cascade of reductions, so we have not pursued it.

C1670 ³⁴<https://www.wolframscience.com/nks/notes-11-8--initial-conditions-for-rule-110/>

C1671 ³⁵An alternative simulation of Neary and Woods [25] simulates a Turing machine directly by a cyclic
 C1672 tag system \mathcal{C}' , and is also much more time-efficient.

C1673 ³⁶Curiously, while the survey of Woods and Neary [45] carefully distinguishes *semi-weak* and *weak*
 C1674 universality, the fundamental characteristic whether the Turing machine has a provision for halting is
 C1675 treated only as an afterthought.³⁷

C1676 ³⁷Incidentally, in Turing's original article [42] from 1937, where the machines that are now called Turing
 C1677 machines were first defined, the good machines are those that *don't* halt or go into a loop (the *circle-free*
 C1678 ones), because they are capable of producing an infinite sequence of zeros and ones on dedicated *output*
 C1679 *cells* on the tape, forming the fractional bits of a *computable number*. The question about his machines
 C1680 that Turing proved to be undecidable is: Does this machine ever print a 0?

C1681

9.3 Wolfram’s 2,3 Turing machine

C1682

C1683

C1684

C1685

C1686

C1687

C1688

C1689

C1690

An even smaller Turing machine with only 2 symbols and 3 states was proposed by Wolfram [44, Section 11.12, p. 709]³⁸ and was shown to be universal, in a certain sense, by Alex Smith [39].³⁹ As above in Section 9.2, the proof performs a reduction from cyclic tag systems, and again, this machine does not halt. Smith showed that the 2,3 Turing machine can simulate cyclic tag systems, but unfortunately, it is not addressed at all what happens when the operation of the simulated cyclic tag system terminates. Besides the issue of halting, there is a more severe obstacle for using this machine for our purpose: The input is not some finite word bounded by repeated patterns on both sides, but an infinite string that has to be set up by an independent process.

C1691

10 Outlook

10.1 Equality testing

C1692

C1693

C1694

C1695

C1696

C1697

C1698

C1699

C1700

C1701

C1702

C1703

C1704

For the reader who has well digested the basic ideas in the two different proof approaches of PFA Emptiness undecidability, it is an instructive exercise to see how Nasu and Honda’s method of testing acceptance probabilities for equality by formula (8) (Section 5.3) would apply to the Equality Checker problem of Section 4.1 for the string $a^i b^j \#$: It is straightforward to set up a PFA with two states that accepts $a^i b^j \#$ with probability $\phi = 1/2^i$, and another that accepts it with probability $\psi = 1/2^j$.

The construction of Figure 6a, translated into the language of the Equality Checker, leads to the following algorithm: The coins are flipped as usual. In the end, when reading the symbol $\#$, the PFA flips two more coins, and

- with probability $1/4$, it accepts if the red coin was unlucky;
- with probability $1/4$, it accepts if the orange coin was unlucky
- with probability $1/2$, it accepts if the blue coin was lucky.

The green coin is ignored. The resulting probability of acceptance is

C1705

$$\frac{1}{4}(1 - 1/4^i) + \frac{1}{4}(1 - 1/4^j) + \frac{1}{2} \cdot 1/2^{i+j} = \frac{1}{2} - \frac{1}{4}(1/2^i - 1/2^j)^2, \quad (27)$$

C1706

C1707

C1708

C1709

C1710

C1711

C1712

C1713

C1714

C1715

C1716

C1717

C1718

which reaches its maximum value $1/2$ if and only if $i = j$. (To save coin flips, one would of course rather take the decision between the three branches in advance.)

We notice a sharp contrast between the character of the outcome in the two cases. The equality test by formulas (8) and (27) capitalizes on the capability of a PFA to detect a tiny fluctuation of the acceptance probability above the cutpoint. On the other hand, the Equality Checker, as illustrated in Figure 2, almost always leaves the answer “Undecided”, but if it makes a decision, the probabilities of the two outcomes, in case of inequality, differ by several orders of magnitude.

We remark that the fluctuation above λ can be amplified by a technique of Gimbert and Oualhadj [15] from 2010, which was considerably simplified by Fijalkow [13]: The Emptiness Problem with strict threshold $> \frac{1}{2}$ can be reduced to the dichotomy “there are acceptance probabilities arbitrarily close to 1” versus “all acceptance probabilities are $\leq 1/2$ ”. See Section 11.3.

C1719

C1720

C1721

C1722

C1723

C1724

C1725

³⁸on-line at <https://www.wolframscience.com/nks/p709--universality-in-turing-machines-and-other-systems/>

³⁹The reader should be warned that the journal version [39] is partly incomprehensible, since the proper horizontal alignment in the tabular presentation of the Turing machines has been destroyed in the typesetting process. Understandable versions can be found elsewhere on the web, see for example <https://www.wolframscience.com/prizes/tm23/TM23Proof.pdf>. Apart from this, the paper could definitely have benefited from some reviewing and editorial guidance.

C1726

10.2 Shortcutting the reduction?

C1727

C1728

C1729

C1730

Figure 11 illustrates the chain of reductions leading to the three undecidability proofs of PFA Emptiness. We have not drawn the detour from the PCP via integer matrices to PFA Emptiness. In all cases, undecidability ultimately stems from the Halting Problem for Turing machines.

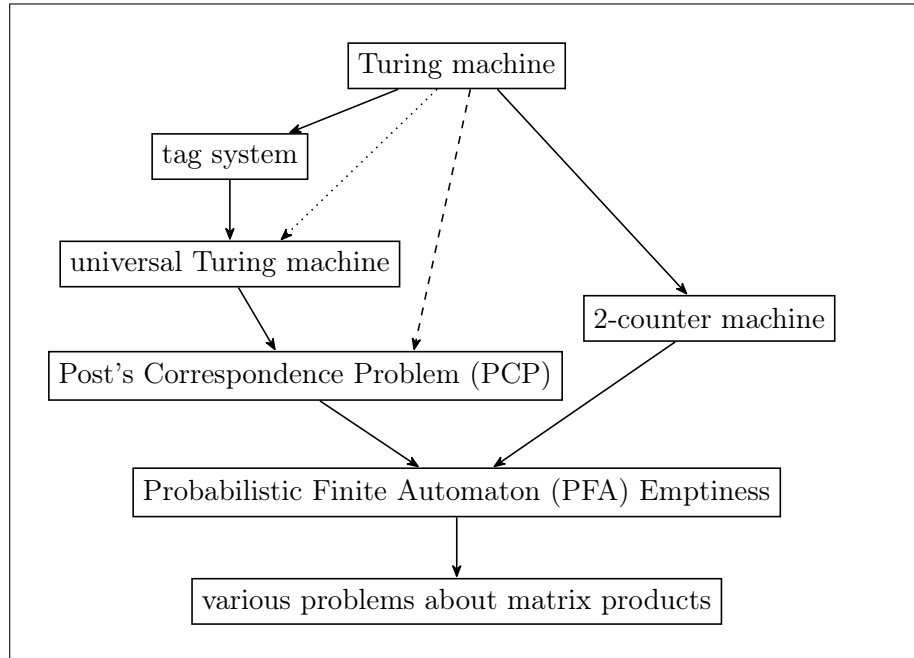


Figure 11: Reductions for proving undecidability. The four topmost boxes concern the *Halting Problem* for the respective systems. The dashed arrow represents the reduction that is sufficient for the plain undecidability result of PFA Emptiness (Propositions 1–4), without the specializations of Theorems 2–4.

C1731

C1732

C1733

C1734

C1735

C1736

C1737

C1738

C1739

C1740

C1741

C1742

C1743

C1744

C1745

The earliest universal Turing machines simulate general Turing machines directly, as indicated by the dotted arrow. However, the smallest universal Turing machines known today do not simulate Turing machines, but tag systems (Section 9.2). In particular, this is true for the machine $U_{15,2}$ of Neary and Woods, on which Theorems 2–4 are based. This has the somewhat curious effect that our construction of specialized undecidable instances of PFA Emptiness proceeds by reduction from tag systems, which operate on *strings*, via universal Turing *machines*, to another problem on *strings*: the PCP. It would seem natural to shortcut this detour and try to go from tag systems to the PCP directly. Tag systems are universal in the sense that every Turing machine can be simulated by some tag system. What might be useful for us is a tag system with a (small) fixed set of rules for which the halting problem is undecidable, depending on the starting string. It seems that such tag systems have not been studied in their own right. Of course, one can take a tag system that simulates a universal Turing machine, an idea that was sketched in Sections 9.1 and 9.2. This adds another round to the detour, but it might be interesting to pursue this approach.

C1746

10.3 The minimum number of states

C1747

C1748

C1749

It is a natural question to ask for the smallest number of states for which the PFA Emptiness Problem is undecidable. The number of states can be reduced to 9 (Theorem 5). When the transition matrices are fixed, the bounds are not much larger: Depending

C1750 on the precise technical formulation of the question, 9 states (Theorem 4a) or 11 states
C1751 (Theorems 2b and 4b) suffice.

C1752 Claus [8, Theorem 7 and Corollary, p. 155] showed that the emptiness question can
C1753 be decided for PFAs with two states.

C1754 11 Epilogue: How to present a proof

C1755 Struggling through the literature and writing this article has prompted me to reflect on
C1756 different ways of presenting things. In this final section, which has become quite long,
C1757 I want to discuss two issues: (1) Choosing the right level of abstraction. (2) Presenting
C1758 material in a self-contained way versus relying on powerful general results.

C1759 11.1 Levels of abstraction

C1760 We have initially defined PFAs by a high-level informal description as an *algorithm*,
C1761 referring to finite-range variables and using metaphors such as flipping of coins. We have
C1762 complemented this with a low-level, formal definition in terms of transition *matrices*.

C1763 An even larger range of abstraction levels exists for Turing machines (cf. [38, p. 144–
C1764 145]). We know that a Turing machine can implement any algorithm. For some task that
C1765 is a straightforward programming exercise, we may just specify at a high level what the
C1766 machine should do, trusting that the reader has internalized the Church–Turing thesis.
C1767 At an intermediate level, we may describe how the Turing machine arranges data on the
C1768 tape and marks cells or carries information back and forth on the tape in fulfilling its
C1769 task. The lowest level is the formal description in terms of the transition function, in
C1770 “Turing machine assembly language”, so-to-speak.

C1771 Each level has its proper place. For example, the description of the PFA for the
C1772 Condon–Lipton proof in Section 4 remains exclusively at a high and abstract level. On
C1773 the other hand, the binary PFA in Section 5.1 is best described in terms of its transition
C1774 matrix. There is nothing to say about its behavior beside the fact that it performs its
C1775 transitions as specified by the matrix.

C1776 People might have different preferences in this matter. In any case, if an informal and
C1777 high-level description is complemented by a more concrete “implementation” at a lower
C1778 level, such a presentation may offer something for every taste. A high-level description
C1779 is suitable for a clearer presentation of the main ideas, which might be obscured by the
C1780 details of a low-level description. Nevertheless, the low-level description may be useful
C1781 to confirm the understanding or to dispel doubts.

C1782 Anyway, an experienced reader will be able to translate between the levels. For
C1783 example, when a machine does several things simultaneously or keeps track of several
C1784 counters, this corresponds to taking the product of the state sets, and the Kronecker
C1785 product of transition matrices (cf. Section 7.2).

C1786 The first two case studies contrast different levels of abstraction.

C1787 11.2 Case study 1: Restricting the output vector f to a 0-1-vector

C1788 In the proof of Theorem 2a in Section 7.5 (p. 34), we convert a PFA with arbitrary
C1789 output values (acceptance probabilities) f from the interval $[0, 1]$ to an equivalent PFA
C1790 with 0-1-values, i.e., with a set of accepting states.

C1791 Let us review this proof from the point of view of the abstraction levels used. After
C1792 stating the idea, we give an informal description of the conversion process. Then we
C1793 describe the process formally in terms of transition matrices and the starting distribution.
C1794 Finally, we make an observation on the resulting matrices, and confirm the understanding

C1795 by interpreting it in terms of the original idea. A formal proof that the new PFA yields
C1796 the same acceptance probabilities is omitted.

C1797 The same statement is proved as the last step of the first proof of Turakainen’s
C1798 more general theorem [40, p. 308] from 1969 that has been mentioned in Section 7 (see
C1799 footnote 23). In this proof, the probabilities are first rescaled to ensure that $\sum f_q = 1$.
C1800 Then the basic idea is the same as in our proof of Theorem 2a: Concurrently with every
C1801 step, we generate a random variable X . In contrast to our proof, this random variable
C1802 has d possible outcomes. Each outcome corresponds to one of the states. Moreover, the
C1803 distribution of X is always the same, independently of the current state: The d outcomes
C1804 of X are chosen according to the distribution f . This variable is sufficient to make a
C1805 decision whether the input word should really be accepted: It is accepted if the value
C1806 of X matches the current state q , which happens with probability f_q . Since there are d
C1807 copies of every state, the number of states increases quadratically.

C1808 The idea, however, is not explained in [40]. Turakainen writes down the $d^2 \times d^2$ tran-
C1809 sition matrix, the corresponding starting distribution, and the accepting set. Correctness
C1810 is proved formally by multiplying out the matrices and showing that they lead to the
C1811 same acceptance probabilities.

C1812 Our own proof in Section 7.5 is not only less wasteful of states, by generating a
C1813 customized random variable X depending on the state, but it also gives explanations
C1814 and some intuition of what one tries to achieve.

C1815 Incidentally, the same statement is proved by Paz with a quite different argument,
C1816 which is very elegant. It appears on [30, p. 151] as part of the proof of a more general
C1817 statement, Theorem 2.4 of Section IIIA. The output vector $f \in [0, 1]^d$ can be represented
C1818 as a convex combination of 0-1-vectors (vertices of the hypercube $[0, 1]^d$). Appealing to a
C1819 general statement about convex combinations [30, Corollary 1.7 of Section IIIA, p. 148])
C1820 concludes the proof. The number of states of the construction is not discussed. As stated,
C1821 the construction may lead to an exponential blow-up of the number of states. However,
C1822 with appropriate extra considerations, for example, by appealing to Carathéodory’s The-
C1823 orem, the blow-up can be reduced to quadratic.

C1824 Turakainen’s second proof [41] from 1975 is much more efficient, see Sections 8.1.2
C1825 and 8.2.4: It uses just one additional state to get to a unit vector f , possibly losing the
C1826 empty word from the recognized language.

C1827 11.3 Case study 2: Probability amplification

C1828 The classical acceptance criterion is the strict threshold $> \lambda$. We will now see a general
C1829 method by which the Emptiness Problem with strict acceptance criterion $> \frac{1}{2}$ can be
C1830 strengthened to the dichotomy “all acceptance probabilities are $\leq 1/2$ ” versus “there are
C1831 acceptance probabilities arbitrarily close to 1”:

C1832 **Theorem 8** (Gimbert and Oualhadj 2010 [15]). *We are given a PFA A with input*
C1833 *alphabet Σ , and we denote its acceptance probability for an input $u \in \Sigma^*$ by $\phi(u)$. We*
C1834 *can construct a PFA B with input alphabet $\Sigma \cup \{\text{end, check}\}$ with the following property.*

- C1835 1. *If every input u for A has acceptance probability $\phi(u) \leq 1/2$, then the same holds*
C1836 *for B .*
- C1837 2. *If A has an input with acceptance probability $\phi(u) > 1/2$, then B has inputs with*
C1838 *acceptance probability arbitrarily close to 1.*

C1839 11.3.1 Algorithm F, Fijalkow

C1840 *First proof of Theorem 8.* We will first describe a simplified construction due to Nathanaël
C1841 Fijalkow [13] from 2017 and show the original proof later.

C1842 To avoid confusion with the acceptance of B , we call the output of the original
 C1843 automaton A “Plus” or “Minus”, accordingly as it arrives in an accepting state or in a
 C1844 nonaccepting state after reading the input.

C1845 The input for the PFA B is structured into *rounds*. Each round processes an input
 C1846 of the form

C1847
$$u_1 \text{ end } u_2 \text{ end } \dots u_n \text{ end check}$$

C1848 with $u_i \in \Sigma^*$ as follows. Each u_i is processed as in A , and when reading the symbol **end**,
 C1849 the machine notes the output of A (“Plus” or “Minus”). The machine flips a coin for each
 C1850 round, and with probability $1/2$, it does one of the following two things when the symbol
 C1851 **check** is read:

C1852 (+) If the output of A was “Plus” for *all* inputs u_1, \dots, u_n of this round, the machine B
 C1853 *accepts* the input once and for all and ignores the remainder of the input. Otherwise,
 C1854 it gets ready for the next round.

C1855 (−) If the output of A was “Minus” for *all* inputs u_1, \dots, u_n of this round, the machine B
 C1856 *rejects* the input once and for all and ignores the remainder of the input. Otherwise,
 C1857 it gets ready for the next round.

C1858 We call the three outcomes of a round ACCEPT, REJECT, and INDECISION. This
 C1859 type of aggregation is similar to the Correctness Test in Section 4.2 (see Figure 3) except
 C1860 that the machine A has no analog of the output “Undecided”.

C1861 If all outcomes were INDECISION at the end of the input and no definite decision
 C1862 has been reached, the machine B rejects the input. The same is true for an incomplete
 C1863 round, i. e., when the last symbol read was not **check**. If the automaton encounters an
 C1864 ill-formed input, where a **check** symbol follows a symbol other than **end**, it also rejects.
 C1865

It is easy to write the probabilities of the outcomes of a round:

C1866
$$\Pr[\text{ACCEPT}] = \frac{1}{2} \phi(u_1) \phi(u_2) \cdots \phi(u_n) \tag{28}$$

C1867
$$\Pr[\text{REJECT}] = \frac{1}{2} (1 - \phi(u_1)) (1 - \phi(u_2)) \cdots (1 - \phi(u_n)) \tag{29}$$

C1868 Figure 12 represents the outcome in analogy to Figure 2 for the Equality Checker of
 C1869 Section 4.1.

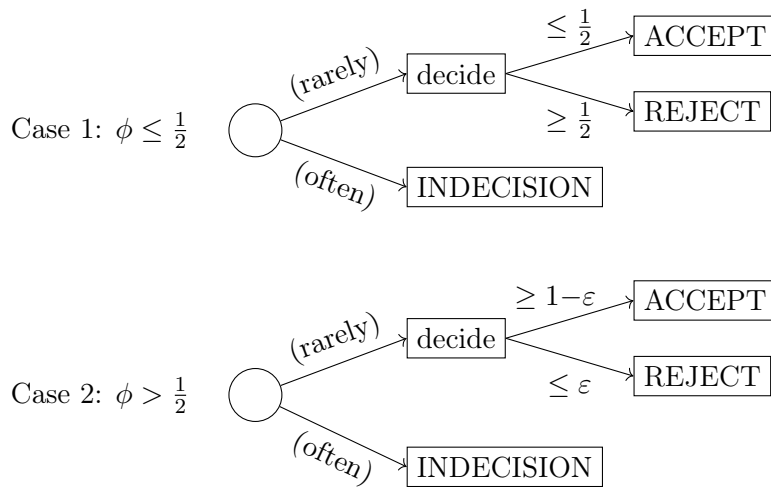


Figure 12: The outcome of processing one round by the PFA B , assuming a large number n of repetitions.

C1870 In case 1 of the theorem, when $\phi(u) \leq 1/2$ for every $u \in \Sigma^*$, the probability (28)
 C1871 cannot be larger than (29). This holds for every round, establishing the first statement
 C1872 of the theorem.

C1873 Let us analyze the other case. Let $u \in \Sigma^*$ be an input for A with $\phi(u) > 1/2$. Then,
 C1874 in the round with n repetitions of this input,

C1875
$$(u \text{ end})^n \text{ check,}$$

C1876 the ratio $(\phi(u)/(1 - \phi(u)))^n$ between (28) and (29) can be made as large as desired by
 C1877 choosing n sufficiently large. We choose n such that

C1878
$$\frac{\Pr[\text{REJECT}]}{\Pr[\text{ACCEPT}] + \Pr[\text{REJECT}]} \tag{30}$$

C1879 becomes less than ε .

C1880 As n increases, the chance of reaching any decision at all become smaller and smaller.
 C1881 We can counteract this effect by trying a large number t of rounds:

C1882
$$((u \text{ end})^n \text{ check})^t \tag{31}$$

C1883 Even if the single probability of reaching a decision, $\Pr[\text{ACCEPT}] + \Pr[\text{REJECT}]$, is very
 C1884 small, the probability of INDECISION in t rounds

C1885
$$(1 - \Pr[\text{ACCEPT}] - \Pr[\text{REJECT}])^t \tag{32}$$

C1886 converges to 0 as t increases.

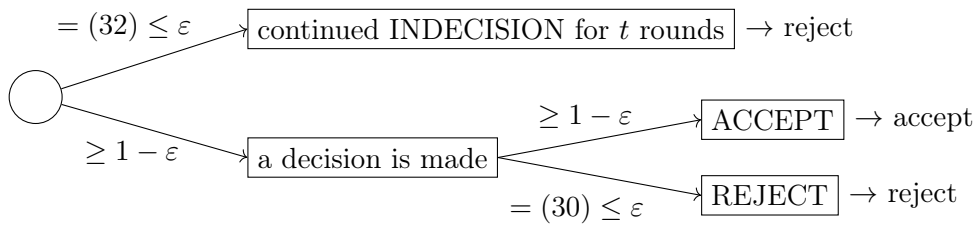


Figure 13: The outcome of t rounds.

C1887 The overall outcome after t rounds is illustrated in Figure 13. The conditional proba-
 C1888 bility (30) of rejection, conditioned on the event that a decision is made, can be reduced
 C1889 below ε by increasing n . The probability (32) of rejection due to indecision can be re-
 C1890 duced below ε by increasing t (depending on n).⁴⁰ In total, the acceptance probability
 C1891 is at least $(1 - \varepsilon)^2$. □

C1892 The procedure is very much reminiscent of the third-level aggregation in the Condon-
 C1893 Lipton proof (Section 4.3) whose idea goes back to Freivalds [14].

C1894 **11.3.2 Algorithm GO, Gimbert and Oualhadj**

C1895 *Original proof of Theorem 8.* The proof of Hugo Gimbert and Youssouf Oualhadj [15]
 C1896 from 2010 constructs a slightly different automaton B . For contrast, we present it here.
 C1897 The input is processed in rounds as before, but acceptance and rejection is decided
 C1898 differently.

C1899 The machine flips a *single coin* at the beginning, and with probability $1/2$, it does
 C1900 one of the following two things:

C1901 ⁴⁰Fijalkow proposes to take $t = 2^n$, and choose n large enough: The sequence of indecision probabilities
 C1902 $(1 - \frac{1}{2}x^n - \frac{1}{2}(1 - x)^n)^{2^n}$ converges to 0 if $x \neq 1/2$.

C1903 (+) If, in some round, the output of A for all inputs u_1, \dots, u_n in that round was “Plus”,
C1904 the machine accepts; otherwise it rejects.

C1905 (−) If, in some round, the output of A for all inputs u_1, \dots, u_n in that round was
C1906 “Minus”, the machine rejects; otherwise it accepts.

C1907 For the analysis, the treatment of Case 1 of the theorem is the same as before.

C1908 For Case 2, let us assume that there is an input $u \in \Sigma^*$ for A with $x := \phi(u) >$
C1909 $1/2$. Due to the single branch at the beginning of the algorithm, the rounds are not
C1910 independent, and the construction of an appropriate input becomes more tricky. Let us
C1911 consider, as before, a number t of identical rounds, each consisting of n repetitions of
C1912 u , as in (31). With $\bar{x} = 1 - x$, we can write the conditional acceptance and rejection
C1913 probabilities in the two branches as follows:

$$\begin{aligned} \text{C1914} \quad \Pr[\text{ACCEPT} \mid (+)] &= 1 - (1 - x^n)^t & \Pr[\text{ACCEPT} \mid (-)] &= (1 - \bar{x}^n)^t \\ \text{C1915} \quad \Pr[\text{REJECT} \mid (+)] &= (1 - x^n)^t & \Pr[\text{REJECT} \mid (-)] &= 1 - (1 - \bar{x}^n)^t \end{aligned}$$

C1916 Keeping n fixed (and large), we can assume that both x^n and \bar{x}^n are close to 0, and we
C1917 see a dilemma: If t is small, the (+) branch will almost always lead to rejection, because
C1918 this is the default result when the input runs out before a decision is reached. If t grows
C1919 to infinity, we will almost always come to a definite decision, no matter how small x^n or
C1920 \bar{x}^n is, and this decision is REJECT in the (−) branch.

C1921 In both of these extreme situations, the acceptance probability will be close to $1/2$.
C1922 We must therefore find the “sweet spot” with just the right number t of rounds. This can
C1923 be done as follows:

- C1924 1. Choose some bound $\varepsilon > 0$ on the rejection probability ($\varepsilon < 1$).
- C1925 2. Set $g := \varepsilon / \ln \frac{2}{\varepsilon}$.
- C1926 3. Pick n large enough such that $\bar{x}^n / x^n \leq g$ and $x^n \leq 1/2$.
- C1927 4. Set $t := \lfloor \ln \frac{2}{\varepsilon} / x^n \rfloor$.

C1928 Then, in the (−) branch

$$\begin{aligned} \text{C1929} \quad \Pr[\text{REJECT} \mid (-)] &= 1 - (1 - \bar{x}^n)^t \leq 1 - (1 - t \cdot \bar{x}^n) = t \cdot \bar{x}^n \\ \text{C1930} \quad &\leq \ln \frac{2}{\varepsilon} / x^n \cdot g x^n = \ln \frac{2}{\varepsilon} \cdot g = \varepsilon \end{aligned}$$

C1931 In the (+) branch, we have

$$\text{C1932} \quad \Pr[\text{REJECT} \mid (+)] = (1 - x^n)^t \leq (1 - x^n)^{\ln \frac{2}{\varepsilon} / x^n - 1},$$

C1933 where the term -1 in the exponent accounts for the rounding of t . We estimate the
C1934 corresponding factor by $(1 - x^n)^{-1} \leq (1 - 1/2)^{-1} = 2$, and get

$$\text{C1935} \quad \Pr[\text{REJECT} \mid (+)] \leq (1 - x^n)^{\ln \frac{2}{\varepsilon} / x^n} \cdot 2 = 2 \left((1 - x^n)^{1/x^n} \right)^{\ln \frac{2}{\varepsilon}} \leq 2(1/e)^{\ln \frac{2}{\varepsilon}} = \varepsilon. \quad \square$$

C1936 The original construction of the input by Gimbert and Oualhadj [15, Section 7 of the
C1937 technical report] is different: They use an increasing sequence n_1, n_2, \dots of round lengths
C1938 in a special way. With this setup, they can reach a small rejection probability by simply
C1939 choosing the number t of rounds large enough.

C1940 11.3.3 The no-coin algorithm NC

C1941 Comparing Algorithms F and GO, we can speculate that Algorithm GO might have its
C1942 origin in a shortage of coins. We find that we actually don’t need any coins at all, as
C1943 shown in the following simplified version of Algorithm F.

C1944 Every round is processed as follows when the symbol `check` is read:

C1945 (±) If the output of A was “Plus” for all inputs u_1, \dots, u_n of the round, the machine B
 C1946 *accepts* the input once and for all and ignores the remainder of the input.

C1947 If the output of A was “Minus” for all inputs u_1, \dots, u_n of the round, the machine B
 C1948 *rejects* the input once and for all and ignores the remainder of the input.

C1949 In case of an empty round or an ill-formed round, the input is also rejected.

C1950 Otherwise, the machine gets ready for the next round.

C1951 We had to come up with a rule for an empty round, because the first two sentences
 C1952 contradict each other in this case. We have decided to reject such an input as ill-formed.
 C1953 Algorithm F would have chosen final acceptance or final rejection with probability $1/2$,
 C1954 but we are not in the mood for making random decisions.

C1955 The calculation goes through as before. The only change is that the factor $1/2$ is
 C1956 removed from (28) and (29).

C1957 11.3.4 High-level verbal description versus state diagrams

C1958 The above description in terms of an algorithm that is formulated in words is not how
 C1959 the PFA B is presented in [13] and [15].

C1960 Fijalkow [13, Figure 1], following [15], starts by showing the *expanding automaton*
 C1961 of Figure 14. This is an abstraction of the automaton B where every simulation of the
 C1962 automaton A (for some input “ u end”) is represented by the symbol `sim` with a probability
 C1963 x of output “Plus”. Two absorbing states \top and \perp represent ACCEPT and REJECT.⁴¹

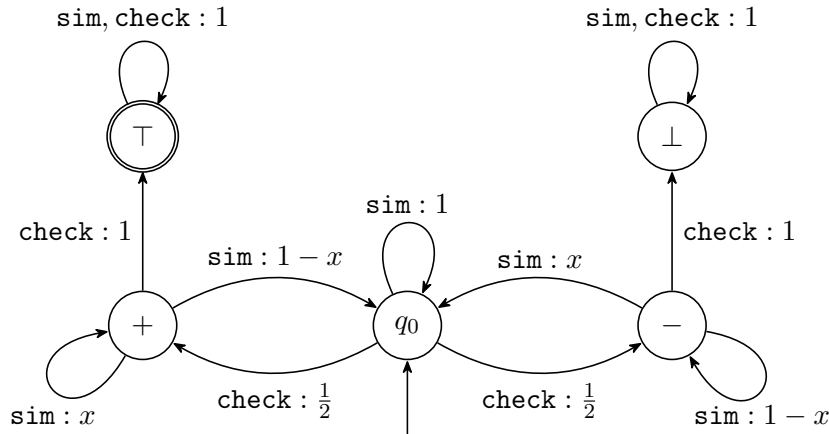


Figure 14: The expanding automaton

C1964 With this interpretation, the behavior of the automaton is very close to Algorithm F,
 C1965 with some insignificant differences. We see that the coin is thrown *before* each round,
 C1966 a detail that was left unspecified in Algorithm F. Consequently, the automaton ignores
 C1967 everything until the first `check` is read, at which point the coin for the first round is
 C1968 thrown. Every subsequent `check` has a dual role: It moves the automaton to one of the
 C1969 two absorbing states \top or \perp when a decision is made, or otherwise it throws the coin for
 C1970 the next round.

C1971 ⁴¹We mention that Fijalkow’s rendition of the expanding automaton [13, Fig. 1] is an improvement over
 C1972 the corresponding automaton in [15, Fig. 1], which has 7 states, not only in representing an algorithm
 C1973 that is easier to analyze: He also chose meaningful symbols `sim`, `check`, and `end` for what is simply called
 C1974 a , b and $\#$ in [15], as well as meaningful names for some states.

C1975 With the expanding automaton, one can very nicely analyze and illustrate the dis-
 C1976 tinction between the cases $x \leq 1/2$ and $x > 1/2$, for which the respective conclusions in
 C1977 Theorem 8 arise.

C1978 The actual automaton B is obtained from the expanding automaton by substituting
 C1979 two copies of the automaton A , identifying the start state of A with the states $+$ and
 C1980 $-$, respectively. When reading the symbol `end`, the automaton makes the appropriate
 C1981 transitions to q_0 , $+$, or $-$, accordingly as the automaton A is in an accepting or rejecting
 C1982 state (for A), and depending on whether we are in the left ($+$) or right ($-$) branch.

C1983 Fijalkow [13, Figure 2] gives a schematic drawing of the automaton B , while Gimbert
 C1984 and Oualhadj [15, Section 8 of the technical report] specify all transitions in detail in
 C1985 words.

C1986 A state diagram may have advantages through its immediate visual appeal. However,
 C1987 I think the difference between Algorithms F and GO can be better explained in verbal
 C1988 terms than by studying state diagrams and abstracting the behavior from it. (Fijalkow
 C1989 2017 [13, p. 15–16] does not explain the distinction but rather downplays the difference.⁴²)
 C1990 Moreover, constructing a PFA for a particular purpose via a state diagram is a tricky
 C1991 business. Indeed, the dual task of the `check` symbol as aggregating the outcomes of the
 C1992 previous round and throwing the coin for the next round leads to an ambiguity whether
 C1993 this symbol (the symbol b in the notation of [15]) should come at the beginning or at the
 C1994 end of a round. Accordingly, some statements in [15, Section 7 of the technical report]
 C1995 are incorrect as written.

C1996 A more significant oversight is shared by [13] and [15]: For a word $u_0 \in \Sigma^*$ that leads
 C1997 back to the start state of A with positive probability, the ill-formed input “`check u_0 check`”
 C1998 is accepted by the automaton B with positive probability, contrary to the intended mean-
 C1999 ing and contrary to what is claimed.

C2000 There are two ways to address this: One simple way is to fix the automaton A
 C2001 by adding a new start state, ensuring that A never goes back to the start state from
 C2002 another state. Another possibility is to extend the analysis by observing that the input
 C2003 “`check u_0 check`” makes equal contributions to ACCEPT and REJECT, and thus it
 C2004 cannot raise the acceptance probability over $1/2$.⁴³ A similar analysis applies to other
 C2005 ill-formed inputs like “`check u_1 end u_0 check`”.

C2006 The reader may ask, why one should be so picky about technical mistakes. Isn’t it
 C2007 more important to convey the main ideas? Well, this precisely underlines the advantages
 C2008 of a high-level verbal description that I am advocating, over an apparently more precise
 C2009 (semi-)formal treatment. If the details are not important, why bother with them? We
 C2010 can just *specify* that the automaton should reject incomplete or ill-formed inputs, leaving
 C2011 out any details about implementing this with states and transitions until someone is
 C2012 interested in the precise number of states.

C2013 Indeed, such a study has been made [35]: If the automaton A has d states, Fijalkow’s
 C2014 automaton B , which has the required properties of Theorem 8 and which carries out
 C2015 Algorithm F with the above-mentioned modifications of behavior, has $2d + 3$ states. The
 C2016 automaton A can have an arbitrary starting distribution, and it can even have output
 C2017 values $f_i \in [0, 1]$. The automaton B will have a single start state and a single accepting
 C2018 state, see [35, Theorem 7 in Appendix A] for a generalized statement with a slightly
 C2019 simplified construction.

C2020 In this respect, Algorithm F has actually an advantage over the simple no-coin al-
 C2021 gorithm NC: NC would need 3 copies of the automaton A : one for the first input u_1 in
 C2022 each round, and two separate copies when the result of the first input is known. Thus,

C2023 ⁴²“The construction is essentially the same as for the undecidability of the value 1 problem by Gimbert
 and Oualhadj [Gimbert and Oualhadj 2010], the main improvements are in the correctness proof.”

C2024 ⁴³Nathanaël Fijalkow, private communication, May 17, 2024.

C2025 the number of states would be $3d$ plus some constant overhead.

C2026 11.4 Case study 3: Coding in binary

C2027 This example and the following case study 4 are about issue (2): Presenting material
 C2028 in a self-contained way versus relying on powerful general results. We look at Lemma 3
 C2029 from Section 7.6 about encoding the input alphabet in binary for input to the PFA.
 C2030 Binary coding is a fundamental and common procedure in Computer Science, and when
 C2031 one thinks about it in terms of a (randomized or deterministic) algorithm, it is a small
 C2032 technical issue that should hardly be worth mentioning. It has been explicitly formulated
 C2033 as a lemma in order to say something about the number of states.

C2034 **Lemma 3.** *Consider a PFA A with input alphabet Σ of size $k = |\Sigma| > 2$, and let*
 C2035 *$\tau: \Sigma^* \rightarrow \{\mathbf{a}, \mathbf{b}\}^*$ be a coding function using the codewords $\mathbf{b}, \mathbf{ab}, \mathbf{aab}, \dots, \mathbf{a}^{k-2}\mathbf{b}, \mathbf{a}^{k-1}$.*

C2036 *Then there is a PFA A' with input alphabet $\{\mathbf{a}, \mathbf{b}\}$ that accepts each word $\tau(u)$ with the*
 C2037 *same probability as A accepts $u \in \Sigma^*$. Words that are not of the form $\tau(u)$ are accepted*
 C2038 *with probability 0.*

C2039 *The number of states is multiplied by $k - 1$ in this construction.*

C2040 The proof of this lemma in Section 7.6 is direct and straightforward. It describes the
 C2041 PFA A' that does the job. The proof is perhaps even a little too much on the verbose
 C2042 side. Writing the two explicit transition matrices is an add-on, to make the description
 C2043 more concrete.

C2044 For comparison, let us look at the treatment in the original sources in Nasu and
 C2045 Honda [23, Lemma 19, p. 269] and in Paz [30, Lemma 6.15, p. 190].

C2046 They appeal to a general statement that acceptance probabilities are preserved under
 C2047 GSM-mappings (mappings induced by a *generalized sequential machine*) [30, Definition
 C2048 6.2 and Theorem 6.10, p. 186]. A sequential machine is a finite automaton with output.
 C2049 The machine is generalized in the sense that, in one step, the machine can also produce
 C2050 several symbols or no output at all.

C2051 Decoding from the binary alphabet $\{\mathbf{a}, \mathbf{b}\}$ to the original alphabet Σ can be easily
 C2052 carried out by a GSM. However, the application of a GSM incurs a complication. More
 C2053 concretely, the encoding function τ in this construction encodes some 7-letter alphabet Σ
 C2054 (see Appendix A) by the binary codewords $\mathbf{a}^i\mathbf{b}$ for $i = 1, \dots, 7$. (Lemma 3 uses a slightly
 C2055 more efficient variant of this code, which is optimized to minimize the number of states.)

C2056 Any GSM that performs the decoding $\tau^{-1}: \{\mathbf{a}, \mathbf{b}\}^* \rightarrow \Sigma^*$ has the undesirable property
 C2057 that “unfinished” strings, whose last codeword is incomplete, are mapped to some word
 C2058 in Σ^* , namely to the decoding of the last complete prefix of the form $\tau(u)$. The same
 C2059 happens for completely illegal words, for example those that contain \mathbf{bb} . This behavior
 C2060 is inherent in the way of operation of a GSM.

C2061 As a consequence, words $x \in \{\mathbf{a}, \mathbf{b}\}^*$ that are not of the form $\tau(u)$ for some u can be
 C2062 accepted with some probability different from 0, contrary to what is required in Lemma 3.
 C2063 (In terms of the algorithm in our proof of Lemma 3, the PFA A' would accept the input
 C2064 based solely on the state q , ignoring the counter i .)

C2065 Thus, in a separate step [23, p. 269, last line], the probabilities have to be patched
 C2066 up to correct this. A finite automaton, and in particular a PFA, can easily check the
 C2067 well-formedness of x , i.e., membership in the regular language $\{\mathbf{ab}, \mathbf{aab}, \dots, \mathbf{a}^7\mathbf{b}\}^*$. Both
 C2068 in [23] and in Paz [30, p. 190, lines 12–13], the correction is expressed very concisely in
 C2069 two lines in terms of a characteristic function χ of this regular language (“event”) and
 C2070 the elementwise minimum operation $g' = g \wedge \chi$. This refers to the intersection $f \wedge g$ of
 C2071 two fuzzy events (probabilistic languages), which is defined in [23, Definition 2, p. 251].

C2072 In Nasu and Honda’s proof [23, p. 269, last line], there is an additional complication:
 C2073 The probabilities have to be patched up in two different ways, because of the more
 C2074 ambitious goal of establishing that the encoded language is an E -set. This requires two
 C2075 PFAs: In one PFA, the ill-formed words are accepted with probability 1, in the other,
 C2076 with probability 0.⁴⁴

C2077 Closure under intersection and union with a *regular* set (or event) is treated by Nasu
 C2078 and Honda in [23, Theorem 4, p. 252] in somewhat obfuscated form, citing their earlier
 C2079 paper [22] from 1968. In Paz, it appears in the same form in [30, Proposition 1.9 of
 C2080 Chapter IIIA, pp. 148–9] with proper credit to [22] (see [30, Section IIIA.3, p. 152]).

C2081 However, when one looks up these statements, one does not directly find the required
 C2082 closure property but something more general. Formulated in our notation, the statement
 C2083 says the following: If $\phi(u)$ and $\psi(u)$ are acceptance probabilities of (rational?) PFAs
 C2084 and the set $\{u \in \Sigma^* \mid \phi(u) > \psi(u)\}$ is a regular language, then $\max\{\phi(u), \psi(u)\}$ and
 C2085 $\min\{\phi(u), \psi(u)\}$ (the union and intersection of two fuzzy languages) are also acceptance
 C2086 probabilities of (rational?) PFAs. Still, the required statement (closure under intersection
 C2087 or union with a regular language) it is derivable as an easy corollary.

C2088 Overall, applying the results on GSM-maps makes the proofs very concise: The proof
 C2089 of Lemma 19 by Nasu and Honda [23, p. 269] takes 17 lines, of which 7 are used for the
 C2090 explicit and detailed specification of the GSM. The proof in Paz [30, Proof of Lemma
 C2091 6.15, p. 190], where the GSM is left to the reader (see footnote 54), consists of only 8
 C2092 lines. However, as we have discussed, the application of the general results about GSMs
 C2093 comes with a considerable conceptual and technical overhead, some of which is hidden
 C2094 by appealing to statements that don’t quite fit.

C2095 There is a final ironic twist: Since we are concerned with the emptiness question, the
 C2096 inaccuracy introduced by the GSM would be of no consequence! If, to a word x that is
 C2097 not of the form $\tau(u)$, we assign the acceptance probability of some other word u' , this
 C2098 does not change the answer to the question whether words with acceptance probability
 C2099 $> \lambda$ exist. In the context of the emptiness question, patching up the probabilities of
 C2100 ill-formed words is unnecessary.

C2101 After this confusing turmoil, the reader may want to go back and savor the directness
 C2102 of the self-contained 13-line proof of Lemma 3 on p. 35.

C2103 11.4.1 Integer matrices

C2104 I want to come back to the first issue, (1) choosing the right level of abstraction, or rather,
 C2105 choosing an appropriate presentation. I will use another instance of binary coding for
 C2106 illustration. I will contrast a representation of transition matrices as block matrices with
 C2107 a representation that defines the entries by a formula via case distinctions.

C2108 **Hirvensalo 2007.** The binary coding technique can be used for products of integer
 C2109 matrices. In this context, the method was used by Hirvensalo [17, Step 3 of Section 3]
 C2110 (see also [16, p. 5]). However, when a number of integer matrices M_1, \dots, M_k is given,
 C2111 the metaphor of Lemma 3 of an algorithm that performs random transitions “according
 C2112 to M_i ” is no longer appropriate. In Section 8.2.3, where the method is described, we
 C2113 have therefore resorted to our next-best choice: a description in terms of block matrices.

C2114 Hirvensalo describes a code that is essentially the same as the code in Lemma 3 and
 C2115 explains the idea of the algorithm that does the decoding as it reads the binary input.
 C2116 The rows and columns of the new matrices are indexed by “states” $(i, q) \in \{0, 1, 2\} \times$
 C2117 $\{q_1, q_2, q_3, q_4, q_5, q_6\}$ (in notation of the proof of Lemma 3).

C2118 ⁴⁴In Paz [30], the same situation occurs at a different place, namely when dealing with deterministic
 C2119 linear languages [30, Proof of Lemma 6.11, p. 188].

C2120 In [17, 16], the transitions are specified very concisely by four formulas that, for each
 C2121 pair of states $(i, q), (i', q')$, gives the entry for row (i, q) and column (i', q') in the matrices
 C2122 M'_a and M'_b , in terms of the entries of the original matrices M_j and a case distinction
 C2123 involving the relation between i and i' .

C2124 There is the added twist that the sixth rows acts like an absorbing state, and we need
 C2125 not bother about continuing the decoding process for the corresponding state q_6 (cf. the
 C2126 remark after Theorem 3 in Section 7.6). This is accommodated in Hirsenshalo's formulas
 C2127 by using only the state $(0, q_6)$ but not the states $(1, q_6)$ and $(2, q_6)$ in the new automaton,
 C2128 thus saving two states. A schematic figure [17, Figure 1] accompanies the description of
 C2129 the decoding process.

C2130 With this type of description, while it is very condensed, it is challenging to make a
 C2131 formal correctness proof, and it would be next to impossible to uncover the mistake in
 C2132 the setup of the final vector f_3 that is mentioned in footnote 31. Here, the block matrix
 C2133 representation of Section 8.2.3 is preferable. It offers a convenient tool to treat compo-
 C2134 sition of transitions, namely, matrix multiplication, and thus it allows us to discover the
 C2135 proper definition of f_3 .

C2136 **Blondel and Canterini 2003.** A simpler but slightly wasteful method for reducing
 C2137 the number of matrices to two was used earlier by Blondel and Canterini [3, Step 2,
 C2138 p. 235]. One matrix, M'_b , is simply a block diagonal matrix of the original matrices M_j .
 C2139 The other matrix M'_a performs a cyclic permutation of the blocks. By a suitable power
 C2140 of M'_a , the “action” can be brought to any desired block of M'_b , so-to-speak.

$$M'_a = \begin{pmatrix} 0 & I & & & \\ & 0 & I & & \\ & & 0 & & \\ & & & \ddots & \\ & & & & 0 & I \\ I & & & & & 0 \end{pmatrix} \text{ and } M'_b = \begin{pmatrix} M_1 & 0 & 0 & \cdots & 0 \\ 0 & M_2 & 0 & \cdots & 0 \\ 0 & 0 & M_3 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & M_k \end{pmatrix}. \quad (33)$$

C2141 The number of states is multiplied by k instead of $k - 1$. This is particularly striking as
 C2142 the main objective of the paper [3] is to get a small number of states.⁴⁵ In any case, the
 C2143 block matrix representation makes the idea very clear, and it is suited for clean proofs.

C2144 11.5 Case study 4: Testing equality

C2145 The equality test requires a PFA with acceptance probability $1/2 - (\phi(a) - \psi(a))^2/4$
 C2146 according to formula (8). In Section 5.3, we have directly the PFA for this particular
 C2147 example. By contrast, the original proof [23, Lemma 11 on p. 259] refers to general
 C2148 closure properties of probabilistic events (languages) under complementation, convex
 C2149 combination, and multiplication [23, Propositions 1–2 and Theorem 3 on p. 252, which
 C2150 are taken from [22]].

C2151 In contrast to the previous example of binary coding, there is not such a clear win for
 C2152 one version of the proof or the other. While the proof in Section 5.3 is direct and does not
 C2153 build on auxiliary results, it talks informally about “mixing” several PFAs, which is just
 C2154 a different way of saying that one forms a convex combination. It introduces the dash-
 C2155 dotted transitions of Figure 6 that are taken before reading the input. The formal details,

C2156 ⁴⁵To be fair, one should remember that the method is used for products of arbitrary, not necessarily
 C2157 stochastic matrices. While the idea that one is just coding the input in another (binary) alphabet comes
 C2158 very natural in a setting where one thinks about an automaton that reads a sequence of symbols, it is
 C2159 not so suggestive when thinking about matrix products.

C2160 In Hirsenshalo [17, 16, Section 4], the method (33) is used in the context of quantum finite automata.
 In this setting, it is the method of choice because the transformation matrices have to be unitary.

C2161 how these transitions can be eliminated and how this is expressed in terms of transition
 C2162 matrices, are also given. The proof in [22, Theorem 4.2] treats convex combinations
 C2163 with a different method: one does not need an extra start state; one can just adapt the
 C2164 starting distribution π . (In our situation, the introduction of the new start state was
 C2165 helpful in order to eliminate the empty word.)

C2166 11.6 Using auxiliary results or starting from scratch

C2167 In a monograph, a textbook, or a longer treatise, it is natural to accumulate a body of
 C2168 techniques and results, as well as notation and abbreviations, on which further results
 C2169 rest.

C2170 Of course, after erecting such an edifice it makes sense to build on it. Such an
 C2171 approach may allow short and potentially elegant proofs. However, if a proof comes
 C2172 out too terse, especially if this is coupled with inaccuracies⁴⁶ and a lack of explanations
 C2173 (What is the main idea? Which parts of the construction are only technical machinery
 C2174 to change acceptance $\geq \lambda$ into $> \lambda$ or to reduce the alphabet size to 2?), it makes a proof
 C2175 practically inaccessible to a reader who is interested only in a particular result, to the
 C2176 point of becoming hermetic.^{47,49}

C2177 In this article, I have enjoyed the freedom of concentrating on a single result and
 C2178 proving everything from scratch. Working with concrete automata in terms of explicit
 C2179 transition matrices has allowed me to come up with the specializations and strengthenings
 C2180 that I have presented.

C2181 References

- C2182 [1] A. Bertoni. The solution of problems relative to probabilistic automata in the frame
 C2183 of the formal languages theory. In Dirk Siefkes, editor, *GI-4. Jahrestagung: Berlin,*
 C2184 *9.–12. Oktober 1974*, volume 26 of *Lecture Notes in Computer Science*, pages 107–
 C2185 112, Berlin, Heidelberg, 1975. Springer-Verlag. doi:10.1007/978-3-662-40087-6
 C2186 _6.

C2187 ⁴⁶Besides the cases discussed above, see footnotes 53 and 54. Also, in [30, p. 190, l. 14], $\tau(g', \lambda)$ should
 C2188 be replaced by $T(g', \lambda)$. This typographical error is disturbing because τ appears frequently in the same
 C2189 paragraph (see Appendix A), whereas $T(g', \lambda)$ is defined in a quite remote location [30, Definition 1.1 of
 C2190 Section IIIB, p. 153].

C2191 ⁴⁷For an experience with the contrary outcome, let me cite the following passage about the philosopher
 C2192 Thomas Hobbes (1588–1679) from his contemporary biographer John Aubrey [12, p. 150]⁴⁸: “He was
 C2193 40 yeares old before he looked on Geometry; which happened accidentally. Being in a Gentleman’s
 C2194 Library, Euclid’s Elements lay open, and ’twas the 47 *El. libri* 1 [the Pythagorean theorem]. He read
 C2195 the Proposition. ‘By G—,’ sayd he (he would now and then swear an emphaticall Oath by way of
 C2196 emphasis) ‘this is impossible!’ So he reads the Demonstration of it, which referred him back to such a
 C2197 Proposition; which proposition he read. That referred him back to another, which he also read. *Et sic*
 C2198 *deinceps* [and so on] that at last he was demonstratively convinced of that truth. This made him in
 C2199 love with Geometry.”

C2200 ⁴⁸<https://archive.org/details/AubreySBriefLives/page/n273/mode/2up>

C2201 ⁴⁹A similar dilemma arises in writing software. Should one use library functions for a particular task,
 C2202 or should one simply program everything from scratch? Using a library often comes with an overhead,
 C2203 because the functions that it offers don’t usually fit exactly and require adaptation.

C2204 There is, however, a crucial difference between writing software and writing mathematics: In pro-
 C2205 gramming, the primary goal is to get things done, as effectively as possible. The target is the computer,
 C2206 and the machine does not have to understand what it is doing, and why. (But yes, software needs to
 C2207 be maintained, and another programmer will have to read and understand the computer code.) Also
 C2208 in mathematics, one wants to get things proved, but the primary target audience is the mathematical
 C2209 reader. Besides convincing the reader of correctness, one also wishes to elicit some understanding.

- C2210 [2] Alberto Bertoni, Giancarlo Mauri, and Mauro Torelli. Some recursive unsolvable
C2211 problems relating to isolated cutpoints in probabilistic automata. In Arto Salo-
C2212 maa and Magnus Steinby, editors, *Automata, Languages and Programming, Fourth*
C2213 *Colloquium. ICALP 1977, University of Turku, Finland, July 18–22, 1977, Proceed-*
C2214 *ings*, volume 52 of *Lecture Notes in Computer Science*, pages 87–94. Springer, 1977.
C2215 doi:10.1007/3-540-08342-1_7.
- C2216 [3] Vincent D. Blondel and Vincent Canterini. Undecidable problems for probabilistic
C2217 automata of fixed dimension. *Theory Comput. Systems*, 36:231–245, 2003. doi:
C2218 10.1007/s00224-003-1061-2.
- C2219 [4] Vincent D. Blondel and John N. Tsitsiklis. The boundedness of all products of a
C2220 pair of matrices is undecidable. *Systems & Control Letters*, 41(2):135–140, 2000.
C2221 doi:10.1016/S0167-6911(00)00049-9.
- C2222 [5] Vuong Bui. Growth of bilinear maps. *Linear Algebra and its Applications*, 624:198–
C2223 213, 2021. arXiv:2005.09540, doi:10.1016/J.LAA.2021.04.010.
- C2224 [6] Vuong Bui. *Growth of Bilinear Maps*. PhD thesis, Freie Universität Berlin, Institut
C2225 für Informatik, 2023. doi:10.17169/refubium-41705.
- C2226 [7] Volker Claus. *Stochastische Automaten*. Teubner Studienskripten. B. G. Teubner,
C2227 Stuttgart, 1971. doi:10.1007/978-3-322-93058-3.
- C2228 [8] Volker Claus. The (n, k) -bounded emptiness-problem for probabilistic acceptors and
C2229 related problems. *Acta Informatica*, 16:139–160, 1981. doi:10.1007/BF00261257.
- C2230 [9] John Cocke and Marvin Minsky. Universality of tag systems with $p = 2$. *J. ACM*,
C2231 11(1):15–20, January 1964. doi:10.1145/321203.321206.
- C2232 [10] A. Condon and R. J. Lipton. On the complexity of space bounded interactive proofs.
C2233 In *Proceedings of the 30th Annual Symposium on Foundations of Computer Science*,
C2234 SFCS '89, page 462–467, USA, 1989. IEEE Computer Society. doi:10.1109/SFCS
C2235 .1989.63519.
- C2236 [11] Matthew Cook. Universality in elementary cellular automata. *Complex Systems*,
C2237 15:1–40, 2004. doi:10.25088/ComplexSystems.15.1.1⁵⁰ URL: [http://www.com-
C2238 plex-systems.com/abstracts/v15_i01_a01.html](http://www.complex-systems.com/abstracts/v15_i01_a01.html).
- C2239 [12] Oliver Lawson Dick, editor. *Aubrey's Brief Lives*. Secker and Warburg, London,
C2240 1949.
- C2241 [13] Nathanaël Fijalkow. Undecidability results for probabilistic automata. *ACM*
C2242 *SIGLOG News*, 4(4):10–17, November 2017. doi:10.1145/3157831.3157833.
- C2243 [14] Rūsiņš Freivalds. Probabilistic two-way machines. In Jozef Gruska and Michal
C2244 Chytil, editors, *Mathematical Foundations of Computer Science 1981 (MFCS)*,
C2245 volume 118 of *Lecture Notes in Computer Science*, pages 33–45. Springer, 1981.
C2246 doi:10.1007/3-540-10856-4_72.
- C2247 [15] Hugo Gimbert and Youssouf Oualhadj. Probabilistic automata on finite words: De-
C2248 cidable and undecidable problems. In Samson Abramsky, Cyril Gavoille, Claude
C2249 Kirchner, Friedhelm Meyer auf der Heide, and Paul G. Spirakis, editors, *Automata*,

⁵⁰This is the DOI according to the web page of the journal. As of 2024-05-01, the DOI link did not work.

- C2252 *Languages and Programming. 37th International Colloquium, ICALP 2010, Bor-*
C2253 *deaux, France, July 2010, Proceedings, Part II*, volume 6199 of *Lecture Notes in*
C2254 *Computer Science*, pages 527–538, Berlin, Heidelberg, 2010. Springer-Verlag. Full
C2255 version in <https://hal.science/hal-00456538v3>. doi:10.1007/978-3-642-141
C2256 62-1_44.
- C2257 [16] Mika Hirvensalo. Improved undecidability results on the emptiness problem of proba-
C2258 bilistic and quantum cut-point languages. Technical Report 769, Turku Centre for
C2259 Computer Science, 2006. URL: http://oldtuucs.abo.fi/publications/view/?pub_id=tHirvensalo06a.
C2260
- C2261 [17] Mika Hirvensalo. Improved undecidability results on the emptiness problem of proba-
C2262 bilistic and quantum cut-point languages. In Jan van Leeuwen, Giuseppe F. Italiano,
C2263 Wiebe van der Hoek, Christoph Meinel, Harald Sack, and František Pláčil, editors,
C2264 *SOFSEM 2007: Theory and Practice of Computer Science*, volume 4362 of *Lec-*
C2265 *ture Notes in Computer Science*, pages 309–319, Berlin, Heidelberg, 2007. Springer.
C2266 doi:10.1007/978-3-540-69507-3_25.
- C2267 [18] John E. Hopcroft and Jeffrey E. Ullman. *Introduction to Automata Theory, Lan-*
C2268 *guages and Computation*. Addison-Wesley, 1979.
- C2269 [19] Omid Madani, Steve Hanks, and Anne Condon. On the undecidability of proba-
C2270 bilistic planning and related stochastic optimization problems. *Artif. Intell.*, 147(1–
C2271 2):5–34, 2003. doi:10.1016/S0004-3702(02)00378-8.
- C2272 [20] Yuri Matiyasevich and Gérard Sénizergues. Decision problems for semi-Thue
C2273 systems with a few rules. *Theoretical Computer Science*, 330(1):145–169, 2005.
C2274 doi:10.1016/j.tcs.2004.09.016.
- C2275 [21] Marvin L. Minsky. Recursive unsolvability of Post’s problem of ‘tag’ and other
C2276 topics in theory of Turing machines. *Annals of Mathematics*, 74(3):437–455, 1961.
C2277 doi:10.2307/1970290.
- C2278 [22] Masakazu Nasu and Namio Honda. Fuzzy events realized by finite probabilistic
C2279 automata. *Information and Control*, 12(4):284–303, 1968. doi:10.1016/S0019-9
C2280 958(68)90353-7.
- C2281 [23] Masakazu Nasu and Namio Honda. Mappings induced by PGSM-mappings and
C2282 some recursively unsolvable problems of finite probabilistic automata. *Information*
C2283 *and Control*, 15(3):250–273, 1969. doi:10.1016/S0019-9958(69)90449-5.
- C2284 [24] Turlough Neary. Undecidability in binary tag systems and the Post correspondence
C2285 problem for five pairs of words. In Ernst W. Mayr and Nicolas Ollinger, editors,
C2286 *32nd International Symposium on Theoretical Aspects of Computer Science (STACS*
C2287 *2015)*, volume 30 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages
C2288 649–661, Dagstuhl, Germany, 2015. Schloss Dagstuhl–Leibniz-Zentrum für Infor-
C2289 matik. doi:10.4230/LIPIcs.STACS.2015.649.
- C2290 [25] Turlough Neary and Damien Woods. P-completeness of cellular automaton Rule 110.
C2291 In Michele Bugliesi, Bart Preneel, Vladimiro Sassone, and Ingo Wegener, editors,
C2292 *Automata, Languages and Programming*, volume 4051 of *Lecture Notes in Computer*
C2293 *Science*, pages 132–143, Berlin, Heidelberg, 2006. Springer. doi:10.1007/11786986
C2294 _13.
- C2295 [26] Turlough Neary and Damien Woods. Four small universal Turing machines. *Fun-*
C2296 *damenta Informaticae*, 91:123–144, 2009. doi:10.3233/FI-2009-0036.

- C2297 [27] Turlough Neary and Damien Woods. Small weakly universal Turing machines.
C2298 In Mirosław Kutylowski, Witold Charatonik, and Maciej Gębala, editors, *Funda-*
C2299 *mentals of Computation Theory*, volume 5699 of *Lecture Notes in Computer*
C2300 *Science*, pages 262–273, Berlin, Heidelberg, 2009. Springer. arXiv:0707.4489,
C2301 doi:10.1007/978-3-642-03409-1_24.
- C2302 [28] Carl V. Page. Equivalences between probabilistic and deterministic sequential ma-
C2303 chines. *Information and Control*, 9(5):469–520, 1966. doi:10.1016/S0019-9958(66
C2304)80012-8.
- C2305 [29] Michael S. Paterson. Unsolvability in 3×3 matrices. *Stud. in Appl. Math.*, 49(1):105–
C2306 107, 1970. doi:10.1002/sapm1970491105.
- C2307 [30] Azaria Paz. *Introduction to Probabilistic Automata*. Computer Science and Applied
C2308 Mathematics. Academic Press, New York, 1971. doi:10.1016/c2013-0-11297-4.
- C2309 [31] Michael O. Rabin. Probabilistic automata. *Information and Control*, 6(3):230–245,
C2310 1963. doi:10.1016/S0019-9958(63)90290-0.
- C2311 [32] Matthieu Rosenfeld. The growth rate over trees of any family of sets defined
C2312 by a monadic second order formula is semi-computable. In Dániel Marx, editor,
C2313 *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA*
C2314 *2021, Virtual Conference, January 10–13, 2021*, pages 776–795. SIAM, 2021.
C2315 doi:10.1137/1.9781611976465.49.
- C2316 [33] Matthieu Rosenfeld. It is undecidable whether the growth rate of a given bilinear
C2317 system is 1. *Linear Algebra and its Applications*, 651:131–143, 2022. arXiv:2201.0
C2318 7630, doi:10.1016/j.laa.2022.06.022.
- C2319 [34] Günter Rote. The maximum number of minimal dominating sets in a tree. In
C2320 Timothy Chan, editor, *Proceedings of the 30th Annual ACM-SIAM Symposium on*
C2321 *Discrete Algorithms (SODA19), San Diego*, pages 1201–1214. SIAM, 2019. doi:
C2322 10.1137/1.9781611975482.73.
- C2323 [35] Günter Rote. Probabilistic Finite Automaton Emptiness is undecidable for a fixed
C2324 automaton, December 2024. URL: [http://page.mi.fu-berlin.de/rote/Papers](http://page.mi.fu-berlin.de/rote/Papers/pdf/Probabilistic+Finite+Automaton+Emptiness+is+undecidable+for+a+fixed+automaton.pdf)
C2325 [/pdf/Probabilistic+Finite+Automaton+Emptiness+is+undecidable+for+a+f](http://page.mi.fu-berlin.de/rote/Papers/pdf/Probabilistic+Finite+Automaton+Emptiness+is+undecidable+for+a+fixed+automaton.pdf)
C2326 [ixed+automaton.pdf](http://page.mi.fu-berlin.de/rote/Papers/pdf/Probabilistic+Finite+Automaton+Emptiness+is+undecidable+for+a+fixed+automaton.pdf), arXiv:2412.05198.
- C2327 [36] Günter Rote. Probabilistic Finite Automaton Emptiness is undecidable for a fixed
C2328 automaton. In Paweł Gawrychowski, Filip Mazowiecki, and Michał Skrzypczak, ed-
C2329 itors, *50th International Symposium on Mathematical Foundations of Computer Sci-*
C2330 *ence (MFCS 2025)*, volume 345 of *Leibniz International Proceedings in Informatics*
C2331 *(LIPIcs)*, pages 8:1–8:18. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2025. to
C2332 appear. URL: [http://page.mi.fu-berlin.de/rote/Papers/pdf/Probabilistic](http://page.mi.fu-berlin.de/rote/Papers/pdf/Probabilistic+Finite+Automaton+Emptiness+is+undecidable+for+a+fixed+automaton.pdf)
C2333 [+Finite+Automaton+Emptiness+is+undecidable+for+a+fixed+automaton.pdf](http://page.mi.fu-berlin.de/rote/Papers/pdf/Probabilistic+Finite+Automaton+Emptiness+is+undecidable+for+a+fixed+automaton.pdf),
C2334 arXiv:2412.05198, doi:10.4230/LIPIcs.MFCS.2025.8.
- C2335 [37] M. P. Schützenberger. Certain elementary families of automata. In Jerome Fox, edi-
C2336 tor, *Mathematical Theory of Automata*, volume 12 of *Microwave Research Symposia*
C2337 *Series*, pages 139–153. Polytechnic Press of the Polytechnic Institute of Brooklyn,
C2338 Brooklyn, N.Y., 1963.
- C2339 [38] Michael Sipser. *Introduction to the Theory of Computation*. PWS Publishing Com-
C2340 pany, 1997.

- C2341 [39] Alex Smith. Universality of Wolfram’s 2, 3 Turing machine. *Complex Systems*,
C2342 29:1–44, 2020. doi:10.25088/ComplexSystems.29.1.1.
- C2343 [40] Paavo Turakainen. Generalized automata and stochastic languages. *Proc. Amer.*
C2344 *Math. Soc.*, 21:303–309, 1969. doi:10.1090/S0002-9939-1969-0242596-1.
- C2345 [41] Paavo Turakainen. Word-functions of stochastic and pseudo stochastic automata.
C2346 *Annales Fennici Mathematici*, 1(1):27–37, February 1975. URL: <https://afm.journal.fi/article/view/134240>, doi:10.5186/aasfm.1975.0126.
- C2348 [42] A. M. Turing. On computable numbers, with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society*, s2-42(1):230–265,
C2349 1937. doi:10.1112/plms/s2-42.1.230.
- C2351 [43] Shigeru Watanabe. 4-symbol 5-state universal Turing machine. *Information Processing Society of Japan Magazine*, 13(9):588–592, 1972.
C2352
- C2353 [44] Stephen Wolfram. *A New Kind of Science*. Wolfram Media, 2002. URL: <https://www.wolframscience.com/nks/>.
C2354
- C2355 [45] Damien Woods and Turlough Neary. The complexity of small universal Turing
C2356 machines: A survey. *Theoretical Computer Science*, 410(4):443–450, 2009. Computational Paradigms from Nature. arXiv:1110.2230, doi:10.1016/j.tcs.2008.09.051.
C2357
C2358

C2359 A The original Nasu–Honda proof in a nutshell

C2360 For comparison, we summarize the proof originally given by Nasu and Honda in [23] and
C2361 reproduced in the monograph of Paz [30, Chapter IIIB, Section 6]. It illustrates how
C2362 people had to struggle with the material when it was new. The presentation of Paz is
C2363 quite faithful to the original, to the point of using identical notation for many notions.
C2364 It is more condensed, slightly simplified, but defaced by the occasional typographic or
C2365 editing error, such as a mysterious passage in a proof that should obviously belong to
C2366 another proof.

C2367 We have adapted the notation to our notation. All references to [30] are to Chapter
C2368 IIIB. (Items in that book are numbered separately in each chapter.)

C2369 **Post’s Correspondence Problem.** The PCP with k word pairs (v_i, w_i) over $\{0, 1\}$
C2370 is represented by the language $L(v, w)$ of words of the form

$$C2371 0^{a_m} 1 \dots 0^{a_2} 10^{a_1} 1 + v_{a_1} v_{a_2} \dots v_{a_m} X w_{b_n}^R \dots w_{b_2}^R w_{b_1}^R + 10^{b_1} 10^{b_2} \dots 10^{b_n}$$

C2372 for sequences $a_1 \dots a_m$ and $b_1 \dots b_n$ with $a_i, b_j \in \{1, \dots, k\}$, where the superscript R
C2373 denotes reversal, see [23, p. 265, before Lemma 16], [30, Lemma 6.13, p. 189]. This
C2374 language is intersected with the set $L_s = \{y + z X z^R + y^R \mid y, z \in \{0, 1\}^*\}$ of palindromes
C2375 with central symbol X and two occurrences of the separator symbol “+” [23, p. 265,
C2376 Lemma 15]. The intersection $L(v, w) \cap L_s$ is well-known to represent the PCP solutions
C2377 [23, p. 270, Lemma 20], because the palindrome property ensures that both the index
C2378 sequences and the produced words match between the two sides. Its emptiness (apart
C2379 from the single word $+X+$) is therefore undecidable [30, Lemma 6.16, p. 190].

C2380 Actually, the alphabet of these languages is $\Sigma = \{0, 1, +, X, \bar{0}, \bar{1}, \bar{+}\}$, because, for technical
C2381 reasons, every symbol σ in the right half, after the X , is replaced by another symbol,
C2382 its “complemented” version $\bar{\sigma}$.

C2383 **Rational automata, P -sets, and E -sets.**

C2384 **Definition 2** ([23, Definition 17, p. 259]). A rational PFA is a PFA where all components
 C2385 of π , f , and the transition matrices $M \in \mathcal{M}$ are rational numbers.

C2386 An E -set is the set of words u with $\phi(u) = \psi(u)$, where $\phi(u)$ and $\psi(u)$ are the
 C2387 acceptance probabilities of two rational PFAs.

C2388 A P -set is a language recognized by a rational PFA with some rational cutpoint λ , or
 C2389 in other words, the set of words u with $\phi(u) > \lambda$.

C2390 We have already seen in Section 5.3 the very important statement that every E -set
 C2391 is a P -set [23, Lemma 11, p.261], [30, Corollary 6.4, p. 183].⁵¹

C2392 **Deterministic linear languages.** The goal is to show that $L(v, w)$, L_s , and finally
 C2393 $L(v, w) \cap L_s$ are E -sets (and therefore P -sets).

C2394 This is done by defining a certain class of context-free languages, the so-called *de-*
 C2395 *terministic linear languages* ([23, Definition 18, p. 263], [30, Definition 6.3, p. 187]), and
 C2396 appealing to the fact that these languages are E -sets [23, Lemma 14, pp. 263–265], [30,
 C2397 Lemma 6.11, pp. 187–188]. The proof relies on m -ary automata.

C2398 It is then easy to see that this language class includes L_s , and also the language $L(v)$
 C2399 of words of the form

$$C2400 0^{a_m} 1 \dots 0^{a_2} 10^{a_1} 1 + v_{a_1} v_{a_2} \dots v_{a_m},$$

C2401 which form the left half of the words of $L(v, w)$, up to the \mathbf{X} symbol. Similarly, the right
 C2402 halves form a deterministic linear language.

C2403 Putting the two halves together to get the language $L(v, w)$ takes more effort. The
 C2404 proof in [23, Lemma 17, p. 265–269] is cumbersome and stretches over more than three
 C2405 pages. Paz formulates the argument as a separate lemma [30, Lemma 6.12, p. 188]:

C2406 **Lemma 4.** *If L_1 and L_2 are deterministic linear languages over disjoint alphabets Σ_1*
 C2407 *and Σ_2 not containing the letter \mathbf{X} , then $L_1 \mathbf{X} L_2$ is an E -set.*

C2408 From the representation of deterministic linear languages as E -sets, we have four
 C2409 m -ary automata: two for L_1 and two for L_2 . The trick is to mix two of these m -ary
 C2410 automata (one for L_1 and one for L_2) into an m^2 -ary automaton: Out of the m^2 digits,
 C2411 one automaton uses the digits $m, 2m, \dots, (m-1)m$; the other automaton uses the digits
 C2412 $1, 2, \dots, m-1$. The acceptance probability is constructed as an m^2 -ary number, but
 C2413 when it is viewed in the m -ary expansion, the digits alternate between the digits for L_1
 C2414 and the digits for L_2 .

C2415 The construction is repeated for the two other PFAs defining the E -sets L_1 and L_2 .
 C2416 Equality of acceptance probabilities then means equality for both L_1 and L_2 .⁵²

C2417 **Intersections of E -sets.** To get from $L(v, w)$ and L_s to $L(v, w) \cap L_s$, one uses the
 C2418 property that E -sets are closed under intersection. This is easy to prove by comparing
 C2419 the acceptance probabilities $\frac{1}{2} - \frac{1}{4}(\phi_1 - \psi_1)^2$ and $\frac{1}{2} + \frac{1}{4}(\phi_2 - \psi_2)^2$, which are constructed
 C2420 along the lines of formula (8) on p. 17 [23, Lemma 12, p. 261].⁵³

C2421 ⁵¹In [30, p. 182], these sets have been renamed to E -events and P -events, respectively. I find this choice
 C2422 of terminology, which goes back to Rabin [31, p. 233] and pervades much of the literature, unfortunate,
 C2423 since an event is rather something whose probability is to be measured. The terms E -language and
 C2424 P -language would have been even more specific.

C2425 ⁵²The requirement of disjoint alphabets is the reason for introducing the complemented symbols $\bar{0}$, $\bar{1}$, $\bar{\cdot}$.
 C2426 Technically, it should not be necessary; any PFA could do this conversion implicitly after the letter \mathbf{X} as
 C2427 it reads the input from left to right.

C2428 ⁵³Paz, in the corresponding passage of his proof [30, Proof of Lemma 6.14, p. 190], appeals to “Exercise
 C2429 4.a.3”, but in that exercise [30, p. 172], closure under intersection is only proved for E -sets that are defined
 C2430 by the condition $\phi(u) = \lambda$ for a constant λ .

C2431 Actually, this intersection property leads to a simple alternative proof of a generalized
C2432 version of Lemma 4, where L_1 and L_2 can be arbitrary E -sets:

C2433 **Lemma 4'.** *If L_1 and L_2 are E -sets over some alphabet Σ not containing the letter X ,*
C2434 *then L_1XL_2 is an E -set.*

C2435 *Proof.* The languages $L_1X\Sigma^*$ and Σ^*XL_2 are certainly E -sets: A PFA can simply ignore
C2436 all symbols before or after the X . Their intersection is L_1XL_2 . \square

C2437 **Coding with 2 symbols.** Finally, the 7-character alphabet $\Sigma = \{0, 1, +, X, \bar{0}, \bar{1}, \bar{+}\}$ is
C2438 converted to binary by a coding function τ that uses the codewords $\mathbf{a}^i\mathbf{b}$ for $i = 1, \dots, 7$.
C2439 We have already seen in Lemma 3, for a very similar code, that this can be done while
C2440 preserving the acceptance probabilities. The step is discussed in detail in Section 11.4.

C2441 Paz appeals to a general statement that acceptance probabilities are preserved under
C2442 GSM-mappings (mappings induced by a *generalized sequential machine*) [30, Definition
C2443 6.2 and Theorem 6.10, p. 186]. (A generalized sequential machine is a finite automaton
C2444 with output over some alphabet Σ . The machine is generalized in the sense that the
C2445 output at every step is from Σ^* , i.e., the machine can produce several symbols or no
C2446 output at all.) Nasu and Honda use their even more general statement about PGSM-
C2447 mappings (mappings induced by a *probabilistic GSM* [23, Definition 13 and Theorem 6,
C2448 pp. 253–255]), which are an object of study in the paper [23] and figure prominently in
C2449 its title.

C2450 Nasu and Honda use this to show that $\tau(L(v, w) \cap L_s)$ is an E -set [23, Lemma 19,
C2451 p. 269]. Paz proves the weaker statement that $\tau(L(v, w) \cap L_s)$ is a P -set [30, Lemma
C2452 6.15, p. 190] and is therefore able to shortcut the proof.⁵⁴

C2453 Since every E -set is a P -set (Section 5.3) and since PFA Emptiness is about P -sets,
C2454 the undecidability of PFA Emptiness is established.

C2455 **A.1 Deciding whether the recognized language is a regular language, or whether it is context-free**

C2456 It is also shown to be undecidable whether a P -set is a regular language, or whether it is
C2457 context-free [23, Theorem 22, p. 270], [30, Theorem 6.17, p. 190]. This can be established
C2458 by the following arguments, which are not given explicitly in [23] or [30], but only through
C2459 unspecific references to the theory of context-free languages.

C2460 If the PCP has a solution, then the language $L(v, w) \cap L_s$ contains some word of the
C2461 form

$$C2462 \quad y+zX\hat{z}\bar{+}\hat{y}$$

C2463 with nonempty $y, z \in \{0, 1\}^*$, where $\hat{y} = \bar{y}^R$ denotes simultaneous complementation and
C2464 reversal. We fix y and z . Since a PCP solution can be repeated arbitrarily, all words

$$C2465 \quad y^{i+z^i}X\hat{z}^i\bar{+}\hat{y}^i \tag{34}$$

C2466 for $i \geq 0$ are also in the language. Moreover, intersection with the regular language
C2467 $\{y\}^* + \{z\}^*X\{\hat{z}\}^*\bar{+}\{\hat{y}\}^*$ leaves *exactly* the words of the form (34), but for such a language

C2468 ⁵⁴This proof has a small technical mistake [30, Proof of Lemma 6.15, p. 190, line 8]: It claims that
C2469 *one can easily construct a GSM mapping Ψ^A to do the decoding.* Some words x are not decodable for
C2470 the reason that they contain more than 7 \mathbf{a} 's in a row or do not end with \mathbf{b} . For these words, it cannot
C2471 be ensured that $\Psi^A(x) = e$ (e denotes the empty word, cf. [30, p. xix, and the footnote on p. 155])
C2472 as Paz claims. The mistake is of no consequence because such ill-formed strings x are filtered out in
C2473 a subsequent step, see Section 11.4. In the original proof of Nasu and Honda [23, p. 269], a GSM for
C2474 decoding is described explicitly.

C2475 it is known that it is not context-free.⁵⁵ Since the intersection of a context-free language
 C2476 with a regular language is context-free, the language $L(v, w) \cap L_s$ cannot be context-
 C2477 free in this case. We conclude that the recognized language is context-free, or regular,
 C2478 (namely, the one-word language $\{+X+\}$) if and only if the PCP has no solution.

C2479 To get the result for a binary input alphabet, this whole chain of arguments must be
 C2480 transferred from $L(v, w) \cap L_s$ to the *encoded* language $\tau(L(v, w) \cap L_s)$, but this does not
 C2481 change the situation.

C2482 There is an alternative proof, following an exercise in Claus [7, p. 158, Aufgabe]. The
 C2483 exercise asks to derive the undecidability of testing whether a given P -set is a regular
 C2484 language as a corollary of the emptiness question. The hint for the solution suggests to
 C2485 take the language L of nonempty solution sequences $a_1 a_2 \dots a_m$ of the PCP, and append
 C2486 some nonregular P -set \tilde{L} to it. This can be done by appealing to Lemma 4'. We have
 C2487 already established that L is an E -set (Section 5.2). Taking some nonregular E -set \tilde{L} , for
 C2488 example the language $\{a^i b^i \# \mid i \geq 0\}$ of Section 10.1, we can form the E -set $L' = L X \tilde{L}$.
 C2489 If the PCP has no solution, L' is empty and therefore regular. If the PCP has a solution,
 C2490 L' is not regular because \tilde{L} is not regular.

C2491 This construction can be extended for context-free languages by taking the language
 C2492 $\tilde{L} = \{a^i b^i c^i \# \mid i \geq 0\}$. It is not context-free, but it is the intersection of two E -sets
 C2493 $\{a^i b^i c^n \# \mid i, n \geq 0\} \cap \{a^n b^i c^i \# \mid i, n \geq 0\}$ and therefore an E -set.

C2494 B Notation, terminology, and abbreviations

C2495 A accepting computation of 2CM, appropriately encoded
 C2496 $A(v, w), \tilde{A}(v, w)$ special integer matrices that model word pairs (v, w)
 C2497 A, A', B PFAs
 C2498 $a_i \dots a_m, b_1, \dots, b_n$ index **sequence** for PCP solution
 C2499 \mathbf{a}, \mathbf{b} symbols for equality checker, alphabet after conversion to binary
 C2500 $B(u)$ binary PFA
 C2501 B_i, C_i, D_i, E_i, F_i successively transformed transition matrices
 C2502 $[0, c_1], (c_1, c_2], (c_2, c_3], \dots$ interval boundaries
 C2503 $d \times d$ size of matrix in joint spectral radius, size of matrix in PFA = **number of states**
 C2504 e_1, e_2, \dots (first) unit vector (local usage)
 C2505 $\varepsilon > 0$. probabilities close to 0 or 1.
 C2506 ϵ empty word (local usage)
 C2507 f, f_q characteristic vector (or more general, output vector) of accepting states q . η^F
 C2508 is used in Nasu and Honda, where F is the set of accepting states. Claus [7] and
 C2509 most others uses f instead of η . Paz p. 150 uses η . Blondel and Tsitsiklis 2000 use
 C2510 0-1-vector η (following Paz).
 C2511 G modulus
 C2512 γ, γ_1 small constant (negative power of 2)
 C2513 Γ tape alphabet, including the blank symbol \sqcup
 C2514 GSM Generalized sequential machine
 C2515 H halting state
 C2516 I unit matrix
 C2517 i, j lengths for equality checker
 C2518 J all-ones matrix, scaled by $1/d$.
 C2519 k number of input pairs for PCP
 C2520 $K = 10$ repetitions until declaring outcome “INCORRECT”

C2521 ⁵⁵In fact, a language like $\{a^i b^i c^i \mid i \geq 0\}$ with just *three* blocks of equal length is the prime example
 C2522 of a language that is not context-free.

C2523	λ cutpoint
C2524	l_i, r_i counter values in 2CM
C2525	L, R movements of TM, as part of the rules (quite local usage)
C2526	MPCP Modified Post Correspondence Problem
C2527	2MPCP doubly Modified Post Correspondence Problem (not used much)
C2528	\mathcal{M} set of matrices
C2529	M, M_i, M_j matrices, transition matrices
C2530	m_1, m_2, \dots, m_d columns of M , (local)
C2531	m length of matrix product in PFA and in joint spectral radius
C2532	m length of PCP solution / length of accepting computation in 2CM
C2533	n length of <i>partial</i> PCP solution
C2534	n length of a round, for amplification
C2535	PFA probabilistic finite automaton
C2536	PCP Post's Correspondence Problem
C2537	p_{00}, p_{01}, \dots transition probabilities in 2×2 automaton $B(u)$
C2538	π, π_q starting distribution
C2539	Q TM states / 2CM states
C2540	$q_i \dots q_m$ states in accepting computation for 2CM
C2541	q, q' states of TM (in rules) / state of PFA
C2542	q_A, q_R extra states for PFA
C2543	RMPCP Reversed Modified Post Correspondence Problem (used only once)
C2544	r_i, l_i counter values in 2CM
C2545	x^R string reversal
C2546	start state, starting distribution, starting pair, starting vector
C2547	$s, s', t \in \Gamma$ tape symbols
C2548	$\sigma \in \Sigma$ (letter from) the input alphabet (of PFA). Currently not used as alphabet of PCP
C2549	Σ output alphabet of a GSM
C2550	t number of repetitions of A to boost the acceptance.
C2551	t number of rounds, for amplification (m in other publications)
C2552	T some Turing machine (local usage)
C2553	TM appreviation for the term "Turing machine"
C2554	$\tau: \Sigma^* \rightarrow \{\mathbf{a}, \mathbf{b}\}^*$ binary encoding
C2555	τ^{-1} decoding, done by a GSM
C2556	u input to TM or to PFA
C2557	$u, u', (v)$ binary strings
C2558	U, U_W, U_0 various universal Turing machines (local usage)
C2559	$0.u$ string u interpreted as a binary fraction
C2560	$(v_i, w_i) \dots (v_k, w_k)$ PCP pairs, pairs of words!
C2561	$x = 0.x_1x_2 \dots x_{47}$ random bits
C2562	$x = \phi(u) > 1/2$, acceptance probability of particular u , for amplification
C2563	y, z STILL MISSING THEIR ROLE!
C2564	Φ, Ψ states
C2565	$\phi = \phi(a), \psi$ acceptance probabilities
C2566	2CM appreviation for the term "2-Counter Machine"
C2567	# separator symbol
C2568	\sqcup blank space