

Michael Rabin's fingerprints across the theory of computation*

Avi Wigderson
Institute for Advanced Study, Princeton†

Abstract

Michael O. Rabin was a major founder and leader of theoretical computer science. He died on April 14, 2026, at the age of 94. This paper aims to survey the wide scope and lasting impact of his research.

Contents

1	Introduction	3
1.1	Preamble: The dawn of computation	4
2	Between decidability and undecidability: algebra	5
2.1	Undecidability of problems on finitely presented groups	5
2.2	Computable algebra	7
3	Finite-state machines and the birth of nondeterminism	9
3.1	Nondeterministic computation	10
3.2	Other modes and variants of finite-state machines	12
3.3	Decision problems about automata	13
4	Between decidability and undecidability: logic, trees and games	14
4.1	Background	14
4.2	Branching computation and Rabin's paper	17
4.3	Infinite games and determinacy	18
5	Computational complexity	18
5.1	Basic notions, choices and goals	18
5.2	The emergence of the class P	19
5.3	Lower bounds	20
6	Probabilistic algorithms	21
6.1	Background	21
6.2	Modeling probabilistic algorithms	22
6.3	Primality testing: randomness for algebra and number theory	24
6.4	Closest pairs: randomness for geometry	26
6.5	Fingerprinting: randomness and communication	27
6.6	Pattern matching: randomness and simplicity	28
6.7	Logic for probabilistic computation	29

*(Random) fingerprints were among Rabin's favorite notions.

†avi@ias.edu

7	Cryptography	29
7.1	Oblivious transfer and secret exchange	31
7.2	Public randomness and beacons	32
7.3	Bounded storage and everlasting security	33
8	Distributed computing	34
8.1	Dining philosophers	35
8.2	Mutual exclusion	37
8.3	Byzantine generals	37
8.4	Information dispersal	39
	Acknowledgements	40
	References	41
A	Complete list of Rabin’s publications and technical reports	58

1 Introduction

The theory of computation, which forms the mathematical underpinnings of computer science, is one of the most important and dynamic disciplines. Since the formalization of the notion of computation by Church, Turing, and Post, the field has constantly expanded its goals and achievements, trying to understand computation, its properties, and its applications in numerous settings and forms. Its models and results have led to remarkable technological revolutions, and have illuminated mathematics and the sciences. Friendly introductions to the field, its diverse parts, and their connections are Harel’s *Algorithmics* [97] (1987) and Wigderson’s *Mathematics and Computation* [241] (2019). Excellent textbooks include Arora and Barak’s *Computational Complexity: A Modern Approach* [9], Goldreich’s *Computational Complexity* [87], Moore and Mertens’ *The Nature of Computation* [160], Papadimitriou’s *Computational Complexity* [171], and Sipser’s *Introduction to the Theory of Computation* [224].

Michael Rabin was one of the giants of this field; in many ways, the evolution of his interests reflected the evolution of the field and shaped many aspects of its development for decades. Rabin described his research path in interviews with Dennis Shasha and David Harel [221, 199], starting with his first encounter with computation. Rabin was a mathematics student at the Hebrew University, focusing on algebra and logic. During his M.Sc. studies at the Hebrew University under Jacob Levitzki, he read Kleene’s book [129] *Introduction to metamathematics*, which described the formative work on computation discussed above. This had a major impact on him. Rabin was also a keen observer of the world around him, and the flurry of activity devoted to building electronic computers compounded that impact. He concluded that the study of computation would lead to new and exciting mathematics, and set out to make good on this prediction. His first step was to go to the heart of this activity and work on his Ph.D. under Church at Princeton, which is the starting point of this story.

The breadth, depth, and impact of Rabin’s work are staggering. The list of areas in which his work was foundational is imposing; it includes computability theory, verification and model checking, automata and formal language theory, computational complexity theory, cryptography and distributed computation.

Papers can be foundational in many ways. Rabin’s papers on computational complexity theory, probabilistic algorithms, and computational algebra created new fields of study, with ample motivation, basic definitions, long-term goals, and initial results. His paper on finite automata with Scott, which earned them the Turing Award, gave the community a template for what it means to study a computational model in depth. His papers in cryptography introduced novel primitives, assumptions, and models. His papers on verification and distributed computing introduced new techniques for solving fundamental open problems in those fields.

Rabin’s distinctive approach reveals itself in his writings. His intellectual pursuit of computational phenomena seeks elegant, simple concepts that capture fundamental issues of lasting significance. His initial works on exploring the power of *nondeterminism* and *randomness* in computation inspire us to this day, and these investigations underlie some of the greatest achievements and open problems of our field. His papers exhibit great vision and powers of abstraction. But despite being a theoretician, Rabin also cared about practice and implementation, and often gave numerical examples of input sizes and runtimes in his papers.

This survey aims to review some of Rabin’s central contributions across many subfields of the theory. It is organized in sections around these subfields, roughly in chronological order. In each, I will try to describe some of Rabin’s work and impact within its historical context. Many of Rabin’s ideas became so ingrained in the field’s culture and in undergraduate education that it may sometimes be hard to appreciate that they had to be created, and how novel they were at the time

of discovery. To illustrate this, I have tried to supply some historical background. The appendix contains what I hope is a nearly complete list of Rabin’s writings, including many technical reports that were never published otherwise.

Some disclaimers are in order. First, this survey is naturally biased by my knowledge, interests, and taste. Second, I focused on readability and accessibility to a general mathematical audience. Thus, in trying to convey the essence of complex frameworks and results, I often oversimplified, neglecting to provide many fine details; I hope this is compensated by the many references to the original texts where the reader can find precise definitions and theorems. These limitations also mean that the survey cannot be a comprehensive historical one, and so does not cover many related works; I hope this is compensated by the books and survey papers I suggested in each section as supplementary reading. In the same vein, there is a large diversity of topics here, and in each there are basic terms that may not be familiar to outsiders; I could not define them all for obvious reasons, but Wikipedia or AI would easily remedy this for the reader. Finally, the history of ideas is fascinating but sometimes also frustrating to explore; it is quite common that ideas arose independently by several people, who may be credited differently in subsequent papers and surveys. I have tried to be as accurate and inclusive as possible in the few relevant cases of ambiguity.

Let me add a brief personal note. I was fortunate to be a colleague of Rabin at the computer science department of the Hebrew University and had an office across the corridor from him for over a decade. One strong impression I had concerns his enthusiasm for research: a few times Rabin burst into my office to tell me about a new idea he had just had, exhibiting the same excitement as a young graduate student with a first discovery. This attitude toward our field and toward research in general was clearly carried over to the classroom, influencing generations of students at the Hebrew University and at Harvard. I feel that this legacy beautifully complements the nature and impact of his work.

1.1 Preamble: The dawn of computation

We start with a very short account of the field before Rabin entered it. Excellent historical accounts (with different foci) are the books of Chabert [35], Davis [50] and Kleene [129].

Like many technological revolutions, the computer revolution had important theoretical foundations. Very early writings on general-purpose “reasoning” go back to Leibniz [142], and writings on actual general-purpose “programming” go back to Lovelace’s work [152] with Babbage on his *Analytical Engine*. But the most direct impetus lay in the crisis in the foundations of mathematics at the turn of the twentieth century. As is well known, Hilbert, perhaps the foremost mathematician of his age, had a dream. His “formalization of mathematics” project was based on two beliefs. First, every true mathematical fact can be proved in an appropriate axiomatic proof system. Second, whether such a proof exists can be determined automatically by a finite “mechanical procedure” [105, 106].

The first belief was shattered by Gödel’s incompleteness theorem [84], which showed that there are true facts about the integers that cannot be proved in Peano Arithmetic. More generally, he showed that every sufficiently strong logical system has statements such that neither they nor their negations follow from the axioms.

The second belief, also known as the *Entscheidungsproblem*, was shattered independently by Church [44, 43], using the lambda calculus, and Turing [236], using his model of machines. Turing’s model, which Church was quick to call the “Turing machine” in a 1937 review [45], mathematically captures Hilbert’s vague notion of a “mechanical procedure”—an essential definition for a mathematical proof that such a procedure does not exist for some task. We will call a function *algorithmically solvable* (and will also use the terms *decidable* or *computable*) if it can be computed by

a Turing machine that halts on every input. That this notion of computability is universal—namely, that Turing machines can simulate *any* other form of computation—is known as the Church–Turing thesis, and has stood unchallenged to this day.

To refute Hilbert’s Entscheidungsproblem, Turing showed that the *halting problem*—the problem that takes as input a description of a Turing machine and asks whether it halts or runs forever, say on an empty input—is unsolvable. Again, we will alternatively use the terms *undecidable* or *uncomputable* below. It is worth pointing out that the halting problem, proved undecidable by Turing, is really a central problem of the nascent field of computer science; every programmer needs to know whether a program is buggy. After Turing, one can ask:

Are there central problems in mathematics that are algorithmically unsolvable?

Suddenly, after Turing’s paper, this question became amenable to mathematical analysis. This possibility had, and still has, a major impact on mathematics. Mathematicians have always been interested in designing algorithms for basic problems. Early examples, all developed millennia ago, include Chinese and Egyptian solutions to linear equations, Euclid’s algorithm for greatest common divisors in the *Elements*, and many constructive proofs in plane geometry. Indeed, the very word *algorithm* derives from a Latinized form of the name of the ninth-century mathematician al-Khwarizmi, whose works described methods for solving arithmetic and algebraic problems. But what about problems that had evaded algorithms? With the mathematical formulation of computation by Turing machines, the game was now afoot to prove such unsolvability results in all areas of mathematics and to discern the boundary between the decidable and the undecidable.

2 Between decidability and undecidability: algebra

This section focuses on two papers of Rabin, both grand in scope, that establish a solid foundation connecting computation and algebra. They exhibit the value of the computational lens even for pure mathematical problems involving no computation. The first [180], from his Ph.D. thesis, contains mainly negative results: basic problems in group theory are proved undecidable. The second [181], presented soon afterward and published in 1960, contains mainly positive results. It is a foundational contribution to the emerging theory of computable algebra, with algorithms and structural theorems for many basic algebraic constructions.

2.1 Undecidability of problems on finitely presented groups

Extensions of undecidability to mathematics followed quickly on the heels of Turing’s result on the halting problem. With the existence of such a natural undecidable problem, to prove that a favorite problem is undecidable one “only” has to reduce the halting problem to it. Suppose the problem is whether an object in a collection of mathematical structures has a particular property. We design an algorithm that translates any given Turing-machine description into a description of such an object, which has the desired property if and only if the given machine halts¹. In other words, we must show that the given problem for such structures can encode general computation.

One of the first mathematical areas to which this idea was applied was group theory. A natural description of a group is by a *presentation*: a set of generators together with a set of relations, namely pairs of words in the generators that are equal as group elements. We assume throughout that presentations are finite, and fix such a presentation. The most basic problem in this setting,

¹More general reductions are possible. For example, it suffices to show that an oracle deciding the target problem could be used by a finite algorithm to decide the halting problem.

and for almost any other algebraic structure, is the *word problem*: given two words in the generators, are they equal as group elements? Algorithms for various special classes of groups were known, most famously Dehn’s algorithm for surface groups [52], but it remained open whether the general problem was decidable.

The first undecidability results came for semigroups, which need not have inverses or identity elements. Their defining relations allow one to replace locally one subword by another, and therefore look similar to the way a Turing-machine configuration changes after one step. Turing machines, however, do not provide the reversibility implicit in equations between words. This hurdle was overcome independently in 1947 by Markov [147] and Post [176], who proved the word problem for finitely presented semigroups is undecidable. Turing subsequently strengthened the result to the more restricted class of semigroups with cancellation [235].

Groups, having inverses and an identity, are more rigid and thus much harder to use for encoding general computation. Nevertheless, Novikov [168] and Boone [24, 25] overcame these obstacles, and proved (via very different constructions) that the word problem for finitely presented groups is undecidable.

Rabin’s *Annals* paper gives a sweeping generalization of this result. To explain in what sense, let us return to Turing machines. Recall that Turing proved that a basic property of a program, namely whether it halts on a given input, is undecidable. Rice [204] extends this undecidability result to every *semantic* property of a program. Here, “semantic” means a property of the partial function computed by a program, and “nontrivial” simply means that the property holds for some such functions and fails for others. Informally, Rice’s theorem means that nothing interesting about the function computed by an arbitrary program can be decided from the program itself.

By analogy, Rabin [180] proves that a vast range of nontrivial properties of groups given by finite presentations are undecidable². Adian obtained closely related results independently [5], and the general result is now called the Adian–Rabin theorem.

Theorem 2.1 (Adian–Rabin [180, 5]). *Every Markov property of finitely presented groups is undecidable.*

A Markov property³ is one that is *nontrivial and semantic*, with similar meanings as above. Here, the word “semantic” is easy to explain: the property should depend only on the group defined by the presentation—not on the presentation itself. In other words, it should be invariant under group isomorphism. The word “nontrivial” is subtler and a bit less trivial to explain: as before, we demand that one group has the property, and another does not, but moreover that the latter is not a subgroup of a group that does have the property⁴.

Rabin, and also Adian, demonstrate the generality of the theorem through many natural examples. In particular, the following group properties are undecidable from a finite presentation: being trivial, finite, cyclic, commutative, solvable, free, and many more. Their techniques also prove other important undecidability results about groups, including the undecidability of testing whether two finite presentations define isomorphic groups. Rabin then draws a direct, purely group-theoretic consequence: there is no complete system of isomorphism invariants that are *effectively computable*.

²For experts; Rabin points out that unlike most previous undecidability results, which were combinatorial in nature, his proof uses algebraic reductions, specifically employing direct product with amalgamation.

³This name was coined soon after by Boone, as both Adian and Rabin follow Markov’s analogous conditions for semigroups.

⁴This “heredity” condition does not present a problem in many applications. To see why, observe that if the word problem (from which the eventual undecidability reduction is made) is solvable for a presentation of some group G , then it is solvable for any presentation of any subgroup H of it. This of course requires that the generators of H can be effectively written as words in the generators of G , something which can be done in many interesting cases.

This strikes a serious blow against natural programs that sought to classify all finitely presented groups by a uniform list of effective algebraic invariants⁵.

The Adian–Rabin theorem made undecidability a general obstruction to classification of groups given finite presentations, and is widely used in combinatorial group theory as a guide to obtaining algorithms and characterizations for various classes of groups. This is one of the earliest examples of this impact of computation on the practice and discourse of mathematics, and it spread quickly to other fields of mathematics. We note that more recently complexity-theoretic notions such as NP-completeness play a similar guiding role in the search for tractable algorithms and characterizations.

2.2 Computable algebra

Algebra has been an algorithmic field from its inception. Some early examples like the GCD algorithm are mentioned in subsection 1.1 and numerous others were added over the centuries for constructing algebraic objects. In these classical settings, however, the algebraic structures usually had explicit descriptions (e.g. integers for GCD) that served as the inputs and outputs of the algorithms.

A major new stage arrived with the 1956 paper of Fröhlich and Shepherdson [77] and Rabin’s 1960 paper [181]⁶. Both initiated and developed the theory connecting computability and algebra, differing somewhat in scope and purpose. Fröhlich and Shepherdson were motivated principally by effective procedures in field theory and developed a theory of *explicit fields*. Rabin aimed at a general integration of computation into algebra and developed a computational theory of abstract algebraic systems, including groups, rings, and fields. Their basic notions are closely related (though not necessarily in a computable way). Rabin explained the reasons for his choices, which, as we shall see, can lead to different results. We begin with common aspects of the two papers and then discuss contributions unique to Rabin’s.

The basic idea in both approaches is simple once a formal notion of computability became available⁷. A countable abstract algebraic system is equipped with an effective presentation: we index the elements of the system by integers (or binary strings), as they would if manipulated by a computer program. We assume that the basic operations in the structure are computable, namely, have algorithms. For a group, composition and inverse operations must be computable on these indices; for a ring or field, addition and multiplication must be similarly computable. Algorithms for other tasks on our algebraic structure can then use these basic operations as subroutines. Conversely, undecidability results in this framework can show that some natural tasks admit no uniform algorithm, and hence fail for some concrete computable systems.

The value of such a theory for developing new algorithmic methods in algebra is clear: an algorithm designed in this setting applies whenever the underlying algebraic structure has the required computable presentation. Both papers develop effective procedures for natural algebraic constructions, including substructures and extensions, and for homomorphisms and isomorphisms between represented structures. They also exhibit failures of effectiveness. Such negative results are equally important: they identify extra hypotheses needed for an algorithm and may guide the search for those hypotheses. Far less obvious is the fact, demonstrated here and in later work, that introducing computation into algebra can also yield new purely algebraic consequences that make

⁵I call this corollary “purely group-theoretic” because such classification programs existed well before computation; the theory of computation offered both a formal definition of what such a classification should look like (e.g., that its invariants are computable) and a way to show that it is hopeless in general.

⁶An early version of Rabin’s work is his July 1957 Cornell Summer Institute notes [182].

⁷Indeed, it follows earlier attempts at such a formalization by Kronecker and van der Waerden, who worked before it was available.

no reference to algorithms in their final statement.

If we abstract the procedures that carry out the operations as oracles, these papers anticipate later black-box algebraic models, which likewise abstract their operations. Theories of black-box groups originated with Babai and Szemerédi [15], and those of black-box fields with Boneh and Lipton [23]. Both focus on *efficient algorithms* rather than finite algorithms, but a prescient remark at the end of Rabin’s paper already suggests replacing the class of computable functions in his theory by any class of algorithms with appropriate closure properties.

Let us conclude with several contributions distinctive to Rabin’s paper [181], drawn from its introduction and its two main parts.

The vision with which Rabin opens the paper is remarkable for such a young researcher shortly after completing his Ph.D. He puts the nascent theory of computability on a par with topology in its possible impact on algebra. Just as one asks whether isomorphisms, homomorphisms, and other constructions involving topological groups are continuous, Rabin proposes asking whether the analogous maps and constructions in effectively represented groups are computable. The same program is equally relevant to rings, fields, and other algebraic systems.

Part I of Rabin’s paper is devoted to a general theory of computable algebraic structures, demonstrated mainly for groups. It contains results about computable homomorphisms, subgroups, and factor groups of computable groups, together with examples showing that the necessary effectiveness hypotheses cannot simply be omitted. Especially striking are the results concerning the word problem. Here the group is already given as an abstract finitely generated group, rather than solely through an arbitrary finite presentation: for a fixed finite generating set, the word problem asks whether a word represents the identity. In a computable presentation this is decidable, since one can evaluate the word. Rabin proves the converse as well.

Theorem 2.2 (Rabin [181]). *A finitely generated group has a decidable word problem if and only if it is computable.*

At this point one might wonder whether noncomputable finitely generated groups exist at all. Rabin obtains them in an elegant way from the Novikov–Boone theorem and Theorem 2.2.

Theorem 2.3 (Rabin [181]). *There exists a finitely generated noncomputable group.*

Theorem 2.2 also has a striking purely algebraic consequence.

Corollary 2.4 (Rabin [181]). *A finitely generated group with an undecidable word problem has no faithful finite-dimensional representation⁸ over any field.*

Assume for contradiction that the group had such a representation, namely were isomorphic to a finitely generated subgroup of $\text{GL}_n(F)$, for some finite n and some field F . Rabin separately proves that such a group *is* always computable, *regardless of whether F is computable or not*⁹. So, by Theorem 2.2 it has a decidable word problem, a contradiction. We note that purely algebraic examples of finitely generated groups with no faithful finite-dimensional matrix representation had previously been constructed by Fuchs–Rabinowitsch [78, 79]; Rabin’s computability-theoretic route is surprisingly direct.

Part II of Rabin’s paper is devoted to computable fields. Its central positive result concerns the basic problem of constructing the algebraic closure of a field.

⁸namely, an isomorphism to a matrix group.

⁹This is a nontrivial existential argument. The key to it is that, since only finitely many elements of F appear as entries of the generators, there exists a computable subfield of F in which all computations can be performed.

Theorem 2.5 (Rabin [181]). *Every computable field F has a computable algebraic closure $\overline{F}$, and a computable embedding of F into $\overline{F}$.*

Rabin points out that one cannot prove this general theorem by simply carrying out the usual Steinitz construction, successively adjoining roots of polynomials through a sequence of finite field extensions. Maintaining an effective presentation along that route would require polynomial factoring algorithms over the intermediate fields, and such algorithms need not exist. Instead, Rabin constructs the closure through a different route due to Bourbaki¹⁰. This route avoids the need for effective factoring algorithms.

The comparison with Fröhlich and Shepherdson is especially illuminating here. They use the Steinitz construction mentioned above, and obtain a different result, in which a closure need not be computable in their sense. For their framework and definitions, a computable algebraic closure exists if and only if the required factoring algorithms exist (which sometimes do not). Rabin’s theorem above proves that in his framework, a computable (in his sense) algebraic closure always exists. For background and subsequent developments, see [227, 153].

To sum up, Rabin’s paper creates a language of computable presentations of algebraic systems and relations between them. It proves when standard algebraic constructions preserve computability, supplies counterexamples showing the limits of those preservation principles, and converts algorithmic questions, such as the word and splitting problems, into structural properties of groups, fields, embeddings, and substructures. Computability (and later, computational complexity) thus became an integral part of algebra.

3 Finite-state machines and the birth of nondeterminism

The previous section dealt with Turing machines, the most general model of computation. In this section we focus on the simplest model, *finite-state machines*¹¹. Indeed, the most basic finite automaton, a *one-way deterministic finite automaton*, abbreviated 1DFA, can be viewed simply as a Turing machine devoid of two central features: the ability to write on its tape, and the ability to reverse direction. Expressed positively, a 1DFA has a finite number of states (some of which are marked “accepting”). It scans its input from left to right, and at every step its current state and the scanned input symbol determine the next state. Thus, the current state—a finite amount of information—is its only memory of what the automaton has seen so far. An input is accepted if the final state is an accepting state. The *language* accepted by an automaton is the set of inputs it accepts.

Rabin and Scott’s paper *Finite Automata and Their Decision Problems* [200] is the centerpiece of this section. It was written in the summer of 1957 at IBM Research. Both were PhD students of Church at Princeton; Rabin had just graduated, and Scott would graduate the following year. The paper earned them the 1976 A. M. Turing Award. The citation for the award summarizes its impact concisely [10]:

This paper introduced the idea of nondeterministic machines, which has proved to be an enormously valuable concept. Their (Scott & Rabin) classic paper has been a continuous source of inspiration for subsequent work in this field.

We shall elaborate on it.

¹⁰For experts, Rabin is working with recursively recognizable maximal ideals in a polynomial ring in countably many variables, using effective root-counting and elimination arguments.

¹¹We will return to Turing machines, and the boundary between decidability and undecidability, in the next section, where we will need the notions of finite automata and nondeterminism discussed in this one.

1DFAs were already studied extensively in the 1950s, and were extremely well understood. Kleene [127, 128] (an early version appeared in 1951) completely characterized their computational power. He introduced *regular languages*¹², and proved that they are precisely those accepted by 1DFAs. Kleene also defined closure properties of regular languages, which led to their concise definition using *regular expressions*. Myhill and Nerode gave structural characterizations of regular languages [166, 167], and of the minimal number of states in an automaton recognizing each one. Moore and Mealy [161, 151] further studied 1DFAs that also produce an output symbol at every step.

With this state of knowledge about the limited power of finite memory, it may be hard to imagine what else can be learned about such machines. The Rabin–Scott paper demonstrated what it really means to understand a computational resource. They initiated a complete study of how different modes of computation and variants of the original model can use finite memory, and the relationships among the language-recognition power and memory usage of these different modes and variants. They also addressed the decidability of basic semantic properties of languages computed by each of these variants, as was done for Turing machines. This comprehensive treatment was foundational for the developing methodology of computational complexity theory.

3.1 Nondeterministic computation

By far, the most important mode of computation introduced in the paper is *nondeterminism*. The state-transition rules of a nondeterministic automaton allow it, given any state and scanned input symbol, to move into any one of several possible states (instead of precisely one in a deterministic automaton). An input is accepted if *there exists* a sequence of such choices that leads to a final accepting state. The resulting model is called a 1NFA, with “N” standing for nondeterminism, here and henceforth. Informally, a nondeterministic automaton is allowed to *guess* the best possible move to make at every step. The main theorem in their paper is that, despite this powerful guessing ability, it does not enable the recognition of more languages. Abusing notation a bit and letting model names represent the classes of languages they accept, we have:

Theorem 3.1 (Rabin–Scott [200]). $1\text{NFA} = 1\text{DFA} = \text{REG}$.

The proof is the famous “subset construction.” The simulating deterministic automaton has a state for every subset of states in the simulated nondeterministic automaton. This construction entails an exponential blow-up in the number of states.

From the outset, Rabin and Scott make several key observations about this new model of computation. First, it can save memory: they state that nondeterministic automata can use far fewer states to compute the same language. They do not prove any lower bound, but a tight exponential lower bound, due to Lupanov [145], soon followed. Furthermore, Rabin and Scott demonstrate how this new mode of computation (together with their theorem) is a powerful descriptive tool allowing simpler proofs and new theorems. One such simple proof establishes Kleene’s closure theorems for regular languages. A new theorem we will soon discuss is the simulation of two-way automata by one-way automata. In both cases, the ability to guess is extremely powerful.

Nondeterministic computation immediately became central to formal language theory. It proved essential for exact machine characterizations of language classes in the Chomsky hierarchy, which Chomsky [38, 39] developed in the 1950s as part of his theory for understanding language. In 1962, Chomsky [40] characterized *context-free* languages (the second level of Chomsky’s hierarchy) as those

¹²Regular languages later featured as the simplest class of formal languages in the Chomsky hierarchy that we will mention in the next section.

recognized by *nondeterministic* pushdown automata¹³. In 1964, Kuroda [132] characterized *context-sensitive* languages (the third level of Chomsky’s hierarchy) as those computed by linear-space *nondeterministic* Turing machines¹⁴.

In parallel, nondeterminism immediately became central to logic and verification, where finite automata on *infinite* objects are studied. We will discuss these at length in Section 4, and mention here only that the central models of this kind—Büchi’s automaton on infinite words and Rabin’s automaton on infinite trees—are nondeterministic.

But the impact on computational complexity is astounding. Complexity theorists have lived and breathed nondeterminism ever since that paper. The natural view of nondeterministic complexity as the complexity of *verification* highlights the fundamental relation between proofs and computations and its many manifestations. Many central complexity classes have an “N” version, and we study its effect on a variety of models and resources; we list just a few examples.

For space complexity we have Savitch’s theorem [213], surprisingly showing that nondeterminism is not so powerful for saving memory, incurring only a quadratic saving over deterministic computation (as long as space is not too tiny, namely $s(n) \geq \log n$).

$$\text{NSPACE}(s(n)) \subseteq \text{DSpace}(s(n)^2).$$

For time complexity, however, we know almost nothing better than the trivial simulation, which incurs an exponential cost when simulating nondeterministic computation deterministically.

$$\text{NTIME}(t(n)) \subseteq \text{DTIME}(2^{t(n)}).$$

For the choice of polynomial time $t(n) = n^{O(1)}$, improving this trivial simulation gives rise to the P versus NP problem—among the most important open questions in computer science and mathematics. In the mid 1960s Edmonds [63, 65] explicitly asked¹⁵ about the relation between the classes P, NP, coNP and $\text{NP} \cap \text{coNP}$. In the early 1970s came the foundational papers of Cook, Karp, and Levin [49, 120, 143] initiating the theory of NP-completeness. These exciting developments surely had an impact on the 1976 Turing award for Rabin and Scott, and the citation for it quoted above.

The guessing power of nondeterministic programs is naturally encoded by an existential quantifier, which leads to connections with logic and suggests further classes obtained by adding a universal quantifier and allowing the quantifiers to alternate, giving rise to the polynomial hierarchy of Meyer and Stockmeyer [157, 226], another fundamental family of complexity classes. Nondeterminism further leads to an alternating-quantifier characterization of PSPACE in [36]. This relation was essential, for example, in Shamir’s proof [216] that $\text{IP} = \text{PSPACE}$. Indeed, the class IP of interactive proofs [14, 93] combines nondeterminism and randomness¹⁶, a development which revolutionized the notion of proofs, creating a rich theory with vast theoretical and practical applications. In yet another direction, alternation of existential and universal quantifiers was added to finite automata by Chandra, Kozen and Stockmeyer [36]. Drusinsky and Harel [59], showed that such automata can be *doubly* exponentially more succinct than deterministic ones (so, another exponential is obtained by adding the universal quantifier to the existential one). More on applying such alternation (and more) to other models can be found in [107, 99].

And the list goes on and on.

¹³Here it is known that the deterministic model accepts strictly fewer languages.

¹⁴Here it is an open question whether the deterministic model accepts strictly fewer languages or not.

¹⁵Using his terminology of “good algorithms” and “good characterizations”.

¹⁶Such alternation is already present in Papadimitriou’s “Games against nature” paper [170] which gives another characterization of PSPACE.

Remark (Turing’s choice machines) One paragraph in Turing’s famous 1936 paper [236] briefly defines what he called *choice machines*, or c-machines, which allow several next moves at any given computational step. He motivates them by proof verification, and distinguishes them from his deterministic *automatic machines*, or a-machines, which are the main focus of his paper. In a later footnote he shows that a finite number of choices can be eliminated in finite time by exhaustive enumeration on a deterministic a-machine¹⁷. This allows him to use c-machines as an intermediate step when simulating the lambda calculus, and then to convert them to a-machines by such exhaustive enumeration.

In short, it is clear that Turing (prescient as ever) defined nondeterministic computation, and viewed it as useful, long before the Rabin–Scott paper. However, this aspect of his paper was apparently not referenced by anyone for decades to come¹⁸. One can speculate on the reasons for this. First, Turing’s paper is so full of fundamental ideas that were crucial for the development of computer science and technology. Second, the cursory manner above in which c-machines occur in the paper does not call attention to a clear separate notion of importance. Finally, it seems that Turing never returned to develop it.

In contrast, Rabin and Scott’s paper highlights this notion, provides the first systematic study of nondeterminism as a computational mode, and demonstrates its utility and power for finite space complexity. This helped make Rabin–Scott the standard source for the language theory and the logic/verification communities (from which the complexity theory community mainly grew). This may explain why the 1976 Turing Award citation credits Rabin and Scott with the introduction of nondeterministic computation.

Interestingly, almost all standard textbooks in complexity theory define the central concept of nondeterministic computation without any reference to its origins. Perhaps this demonstrates how obvious the notion has become to everyone. Given the above, I feel that Turing should be credited with the first definition of this concept, and Rabin–Scott (who, like everyone at the time, did not seem to know about it) should be credited with its reinvention, development, and impact.

3.2 Other modes and variants of finite-state machines

Rabin and Scott did not stop with nondeterminism. They used finite automata as a setting in which to compare several modes of computation.

Two-way automata. A 1DFA reads the input from left to right. A 2DFA may move its input head both left and right, like a Turing machine. This seems like a significant increase in power: the machine can revisit earlier scanned symbols, and thus use the input itself as a kind of read-only workspace. Nevertheless, Rabin and Scott prove that two-way motion does not increase the class of recognized languages.

Theorem 3.2 (Rabin–Scott [200]). $2DFA = 1DFA = REG$.

As noted above, the simulation proof of 2DFAs by 1DFAs is in two steps: first, a simulation of a given 2DFA by a 1NFA, fully utilizing the power of nondeterminism, followed by a 1DFA simulation of that 1NFA using the theorem above. Again, the simulation can be costly in states, namely in memory. Thus one-way and two-way automata have the same decision power, but may differ sharply in memory complexity. Indeed, later work of Kapoutsis [118] proved that in this simulation there are tight exponential upper and lower bounds on the blow-up.

¹⁷Needless to say, Turing cared only about finite algorithms and not their complexity, so the cost of this exhaustive enumeration was immaterial to him.

¹⁸A 1985 book review by Israel [114] identifies Turing’s c-machines with nondeterministic Turing machines.

Two-way nondeterministic automata. Nondeterminism and two-way motion can be combined, resulting in 2NFAs. Again, Rabin and Scott prove that the resulting model still recognizes only regular languages.

Theorem 3.3 (Rabin–Scott [200]). $2\text{NFA} = 2\text{DFA} = \text{REG}$.

Again, the simulation of 2NFAs by 2DFAs can result in an increase in the number of states. But here the quantitative bounds are far from matching. We have an exponential upper bound, but only a quadratic lower bound on the blow-up in the number of states (see the works [211, 41, 42, 4]). Proving a superpolynomial lower bound remains a challenging problem¹⁹.

Multitape automata. In all models considered so far, finite automata had only one tape. Rabin and Scott now introduce and study finite automata with two or more tapes. Here the story changes dramatically. The addition of multiple tapes can create genuine increases in power. In particular, two-way two-tape automata cannot in general be simulated by one-way two-tape automata. Indeed, the paper gives an explicit non-regular language²⁰ which *can* be accepted by a two-way automaton with two tapes! This provides an important counterpoint to the equivalence theorems: some changes in the finite-state model preserve decision power, while others do not.

Probabilistic automata. In a later paper [184], Rabin explores yet another novel mode of computation, adding *random transitions* to finite-state automata. Being a major initial work in the study of randomness as an algorithmic tool and computational resource, we defer a fuller discussion to Section 6.

3.3 Decision problems about automata

The title of the Rabin–Scott paper emphasizes not only finite automata, but their decision problems. So far we have described their work on the decision problems *solved* by finite automata. Their comprehensive treatment now turns to decision problems *about* finite automata: problems where the input is a finite automaton, and we seek information about the language it computes. Analogous to similar work for Turing machines, Rabin and Scott turn to consider which natural semantic properties of finite automata are decidable. Typical examples include *emptiness*, which asks whether the language computed by a given automaton is empty, and *equivalence*, which asks whether two given automata compute the same language.

As above, there is a sharp transition between the difficulties of such problems when they concern one-tape finite automata and those concerning multitape ones. On the one hand, the paper proves that for all models of one-tape finite automata mentioned above, these and related problems are decidable. Indeed, this tractability makes these models amenable to analysis, allowing a useful theory of these models. Of course, these results build on the extensive understanding of the structure of one-tape finite automata developed before and in their paper, and the equivalences they proved between them. On the other hand, they show that even the emptiness problem for two-tape, two-way DFAs is undecidable.

Theorem 3.4 (Rabin–Scott [200]). *It is undecidable to decide whether a given deterministic two-tape, two-way finite automaton accepts any input.*

¹⁹This would not contradict Savitch’s theorem above, as here space $s(n) = O(1)$ whereas there it must be $s(n) \geq \log n$.

²⁰This is really a *relation* on input pairs on the two tapes that are accepted. A modern discussion of this issue can be found in [80].

This part of the paper connects back to the previous section and, as we shall soon see, forward to the next one. Rabin and Scott continue to explore the fine line between decidability and undecidability, and its dependence on delicate differences between the exact details of their input structures.

For further reading on automata and formal-language theory, see Eilenberg [66], Hopcroft and Ullman [111], Kozen [131], and Sakarovitch [210].

4 Between decidability and undecidability: logic, trees and games

This section is devoted mainly to Rabin’s paper [186], which proves the decidability of an important logical theory. In several late interviews, Rabin describes this paper as the most difficult work he ever did [221, 199]. We will now slowly develop the background needed to explain Rabin’s result, and then discuss some of its consequences. Different perspectives on this evolution can be found in the surveys of Thomas [231, 230], Grädel–Thomas–Wilke [94], and Vardi–Wolper [240].

4.1 Background

Recall that Hilbert’s *Entscheidungsproblem* calls for an algorithm to determine whether a given mathematical statement is valid [106]. Of course, to make the question formal, we must fix an underlying proof system.

The most basic logical theory is the so-called Peano arithmetic. It is a first-order theory of the natural numbers with addition and multiplication, with familiar axioms and deduction rules. Many theorems in large parts of mathematics are proved in this system. It was thus especially devastating that, in the negative answer to Hilbert’s *Entscheidungsproblem*, the work of Gödel, Church, and Turing proved that, for this very theory, testing validity is undecidable [84, 43, 236]. The same holds for testing the *provability* of a given statement²¹.

Indeed, Turing proved undecidability for any *sufficiently strong* logical theory, where this term informally means that the theory can argue about computation. For any such theory, there is no algorithm to decide whether a given formula is valid, or if it is provable. This naturally leads to the question: *Which logical theories are decidable?* We will see in this section how different changes in either the logic or the underlying structure can completely change the answer.

In 1929, Presburger [178] proved that the first-order theory of the natural numbers with addition but *without* multiplication is decidable. This so-called Presburger arithmetic is quite strong and can prove many statements about “linear structures” within the integers. But the absence of multiplication allowed Presburger to develop an algorithm for the validity of any formula. Note that this result predates Gödel’s incompleteness theorem, and was a positive result within Hilbert’s program.

Another major decidability result was obtained in 1951 by Tarski [229]. He proved that the first-order theory of the real numbers with addition and multiplication is decidable. Thus, replacing the integers by the reals restores decidability even while retaining both field operations. To prove his result, Tarski developed “quantifier elimination,” a central algorithmic technique.

The study of the two theories above was motivated by mathematical considerations: understanding Diophantine problems about natural numbers and about systems of polynomial equations and inequalities in real-closed fields, respectively. The study of the following theories was motivated

²¹By Gödel’s *completeness* theorem, validity and provability are the same for most proof systems in use, but anyway their undecidability proof applies to both without using that theorem. When talking about the undecidability of a proof system, we would usually mean that testing validity is undecidable, but may switch to testing provability.

by computer-science considerations. This program, connecting logic and language theory with computation, was initiated by Church [46] with his *analysis* and *synthesis* questions. The analysis question asks us to determine the properties, expressed logically or syntactically, of the input–output behavior of computer programs. The synthesis question asks us to design a program whose input–output behavior meets specifications so expressed. For both, we seek the relation between the expressive power of the logical theory describing the properties and the computational power of the programs or machines that realize them. This study has remained important throughout the history of computer science, as computational models evolve and ensuring that what is specified is what is implemented—in hardware, software, protocols, and now AI models—remains essential to the field.

This program was met with spectacular success in the following years within formal language theory (which we mentioned in the previous paragraph and is worth repeating here). The first major success of this program was due to Kleene [128] (who also coined the terms “analysis” and “synthesis”). He defined *regular languages* and proved that they are precisely the languages computed by one-way deterministic finite automata. Then Chomsky [40] characterized the richer class of *context-free languages*²² as those computed by *nondeterministic pushdown automata*. And then Kuroda [132] characterized the even richer class of *context-sensitive languages* as those computed by nondeterministic linear space Turing machines. In all cases, the equivalences (analysis and synthesis) were algorithmically constructive.

But another main focus in this area, in work following Church in the late 1950s by Elgot, Trakhtenbrot, and Büchi [233, 67, 30], was the development of different frameworks for specifying desired properties of computations, and studying the computational models they capture. One difference is that the specifications in the above results are *language-theoretic* (a grammar for the language), while these new works used *logical* specifications (which we will describe after discussing the models). Another difference is the emphasis on properties that hold throughout the computation. Many computer systems²³ interact with their environment: operating systems, for example, repeatedly receive input signals from processes that request a resource (or raise an alarm or effect a change), and respond with some output action. For such systems, especially ones we design, we typically impose requirements on the relation between input and output that should be satisfied throughout.

A natural abstraction of this very general scenario is called, in modern language, a *reactive system*, defined in this generality by Harel and Pnueli [98]²⁴. In reactive systems, the inputs I are words over some alphabet, and the responses R are words over the set of states of our system. At time step t , an input symbol I_t arrives, and the system, given its current state, processes it and moves to a new state, R_t , in a manner consistent with its transition rules. We may now specify, for example, using logical formulas, the sets of input–output pairs of words (I, R) that satisfy the requirements. A possible output R is usually called a *run*, as it describes the behavior of the system as we “run” it on input I .

What should be the weakest formal theory needed to specify natural properties of pairs (I, R) ? The discussion below hints at the art of navigating the delicate balance between the expressiveness of a logical theory and the complexity of deciding it. Clearly, a first-order formula over the integers can specify consistency of the local transition rules at every step (for both deterministic and nondeterministic computation). For finite computations, it can check that the final state is accepting.

²²Essential for the developing theory of programming languages.

²³And, with hindsight, also biological systems (for example, animal behavior) and societal systems (for example, emergency services).

²⁴In the original papers such systems have different names, including sequential circuits, logical nets, and Moore automata. A very related modern term (typically with different applications in mind) is the notion of an *online algorithm*.

Moreover, as the only “arithmetic” in action is repeatedly shifting time ahead by one, we do not even need addition (as in Presburger arithmetic), and can instead allow only the “add one” successor function, $s(t) = t + 1$.

However, first-order statements in this setting do not suffice to check most interesting properties of computations. Basic properties of reactive systems as above, for example, “every request is eventually satisfied,” require quantification over *subsets* of time steps. First-order theories allow only quantification over elements of the underlying structure (which for time steps, can be taken to be the integers). Theories allowing quantification over predicates over the universe are called *second-order*, and are vastly more powerful. Still, an important observation was that most desired properties²⁵ need only quantifiers over *unary* predicates, namely over subsets of the universe. Such theories are called *monadic second-order* theories, which are still very strong. Can we expect decidability of such strong theories?

The decisive positive answer was given in Büchi’s 1962 paper [31]. He proved decidability of S1S, the monadic second-order theory over the integers with the successor function.

Theorem 4.1 (Büchi [31]). *S1S is decidable.*

Büchi’s proof has several remarkable features. First, he notes that, as S1S can argue about subsets of integers, it can argue about infinite words. This leads him to extend the usual model of finite automata (whose inputs are finite words) to operate on infinite words. Specifically, *Büchi automata* are one-way, *nondeterministic*²⁶ finite automata whose inputs are *infinite* words. When does such an automaton accept an input? For a finite input, we accept if the final state is accepting. But for an infinite input, the automaton works forever and never reaches a final state. A Büchi automaton is defined to accept an input I (an infinite word) if some possible run R on I enters an accepting state infinitely often. Observe that this condition can easily be expressed in S1S. Büchi automata are sometimes called ω -automata, and languages over infinite words are sometimes called ω -languages.

Next, Büchi proves an equivalence between logic and automata. To every S1S-formula there corresponds a Büchi automaton such that the formula is valid if and only if the automaton accepts *all* inputs. More generally, the satisfying assignments of the formula can be made to be exactly the language accepted by the automaton. The proof of this theorem proceeds by a natural induction on the structure of the formula. The second-order existential quantifier corresponds to projection, negation is complementation, and the connectives, conjunction and disjunction of formulas become intersections and unions of languages. By far, complementation is the hardest to achieve, as can be expected from the nature of the accepting condition²⁷.

Finally, and this is the main novelty in Büchi’s paper, the equivalence above is used to prove his computability theorem. The algorithm for testing the validity of an S1S-formula first constructs the associated automaton, and then checks that it accepts all inputs. This idea was a methodological transformation affecting logic, automata theory, and program verification. We note that the complexity of this algorithm, while finite, is nonelementary, which Meyer [156] proved is unavoidable.

²⁵For example, that no error ever occurs, that no processor “starves” forever, that every packet is eventually delivered, etc.

²⁶This choice is essential, and in the setting of infinite words deterministic Büchi automata are strictly weaker. However, McNaughton [150] later showed that nondeterministic Büchi automata can be simulated by deterministic *Muller* automata, a variant of Büchi’s. This turned out to be important in Rabin’s proof. An exponentially faster determinization was achieved by Safra [208]; this paper had major impact on the field of verification.

²⁷Interestingly, this argument uses Ramsey theory in an ingenious way which allows the complement of a language to become a *finite* union of (simpler) languages.

4.2 Branching computation and Rabin’s paper

This result of Büchi became one of the foundations of the now vast fields of verification and model checking, capturing many important properties of *linear* computation, like safety, liveness and fairness. But even Büchi himself was not completely satisfied, and asked what could be done about *branching* computation [31]. In such a computation, the input resides on the vertices of, say, a binary tree, rather than on a linear path. A possible run is again a labeling of the vertices by states of the system. Computation by an automaton is still local: transition rules relate the input I_v and the state R_v at any node $v \in \{0, 1\}^*$ to the states labeling its two children, R_{v0} and R_{v1} . The accepting condition should be satisfied for every infinite path down from the root.

Observe that a theory describing properties of such branching computations would be far more expressive. Specifically, it would be able to address, at once, all possible behaviors of the environment interacting with our system (whereas in linear systems the environmental behavior is encoded in the input). In other words, it allows specification over all possible futures. The analogous logical theory, the monadic second-order theory of *trees* (as opposed to words), is a simple modification of S1S: one allows *two* successor functions instead of one, $s_0(v) = v0$ and $s_1(v) = v1$, and it is appropriately called S2S. The decidability of this powerful theory was posed as an open question by Büchi in his paper [31].

Rabin solved this problem.

Theorem 4.2 (Rabin [186]). *S2S is decidable.*

As with Büchi, Rabin proves this via an equivalence between formulas and automata. To every formula in S2S, Rabin constructs a finite automaton over binary trees, which yields a correspondence between satisfying assignments of the formula and inputs accepted by the automaton. This *Rabin tree automaton* is again a nondeterministic finite automaton, and on any input labeling tree vertices, a possible run labeling them must satisfy local consistency of the transition rules for every vertex and its two children. What becomes far subtler is defining an accepting condition. Rabin puts a condition on every branch (infinite path down from the root of the tree). Fixing finitely many pairs of subsets of the automaton’s states, (E_i, F_i) , *Rabin’s acceptance condition* demands that along every branch there is an index i such that the run visits some state in F_i infinitely often, but visits every state in E_i only finitely many times. This delicate choice is critical for the equivalence and the algorithm. Both follow the structure of Büchi’s proof, namely induction on formula structure, but implementing them is far more complex. Indeed, to accept an input, there must exist a run in which every branch is accepting. This requires a very different complementation procedure for which Rabin invents new tools. His paper is considered a technical tour de force.

The impact of Rabin’s paper was enormous, and we conclude by discussing some of it. The insightful definitions and new techniques Rabin developed became essential tools in the field of verification and model checking. Most consequences of the main result are due to the strength of a theory on branching structures. Rabin’s paper already demonstrates this by giving, as corollaries, new decision procedures for an impressive list of logical theories (including ordered, arithmetic, topological, and lattice-theoretic theories), by embedding them all in S2S. The expressivity of the branching structure greatly expands the set of computational properties that can be captured by a decidable theory. As mentioned, branching allows us to consider at once all eventualities caused by the environment (including, for example, adversarial behavior, timing issues, orderings of actions by multiple processes, and more), and formulas can express relationships between different branching futures. Rabin’s theorem influenced many developments of other logics, such as modal and temporal logics, and decision procedures for them. Surveys illuminating various aspects of this impact include [231, 230, 94, 240].

4.3 Infinite games and determinacy

We conclude with another advantage of having tree structure: it allows us to model and study infinite games. At a conceptual level, the verification and model checking literature often views synthesis as a game played between the program and the environment, where a winning strategy for the program means meeting the given specifications. But Rabin also shows concrete corollaries for the theory of infinite games. This theory, initially developed by Gale–Stewart [81] and continued by Wolfe [242] and Martin [148], extends *determinacy* theorems to larger and larger classes of infinite games. These results extend Zermelo’s determinacy theorem for finite win–lose two-player games [246], which states that some player always has a winning strategy. Rabin’s paper shows how Wolfe’s theorem follows easily as a corollary from the decidability of S2S. Interestingly, Rabin had already made the following important point about this theory in an earlier paper [179]. He gives simple examples of infinite games that are determined, namely, one of the parties always has a winning strategy, but none of that player’s winning strategies is computable. Thus, the utility of having a winning strategy is doubtful, and this result calls for investigating the existence of computable ones when defining determinacy.

5 Computational complexity

Computational complexity is today one of the most exciting and influential intellectual disciplines, with great impact not only on computer science, but also on mathematics, the exact and social sciences, and technology. Part of its success depended on extremely prescient choices made when it was born in the 1960s, choices that were central in allowing its initial structures, goals, techniques, and results to emerge. These were then refined and expanded repeatedly to yield the theory and applications we have today.

The centerpiece of this section is Rabin’s 1960 paper, *Degree of Difficulty of Computing a Function and a Partial Ordering of Recursive Sets* [183], in which he initiated a general computational complexity theory. In it he lays out and motivates, in clear language, many of the important choices and focal points that we now take for granted. We discuss these foundational issues first. When reading this part, try to imagine a world when quantitative analysis of algorithms hardly existed.

Remark 5.1. *We note that this paper was only published as a technical report of the Hebrew University. However, we know that these ideas have influenced later papers that were foundational to complexity theory, including those of Cobham [47], Ritchie [205], and Hartmanis–Stearns [103]. Cobham cites Rabin’s paper in his. Ritchie’s paper acknowledges conversations with Rabin. Hartmanis, in his memoir on the origins of computational complexity theory [101], tells how talking to Rabin when he and Stearns were writing their paper sent them to “eagerly read his manuscript.”*

5.1 Basic notions, choices and goals

Complexity matters. Church, Turing, and Post initiated modern computability theory by establishing a fundamental fact about computation: some natural problems are uncomputable. But this is only a starting point. Not all computable functions are born equal, and having a finite algorithm may be nearly as useless in practice as having none. Rabin opens his paper with an example of two functions, one that is trivial to compute (it takes linear time), and one that seems much harder (the only known algorithm takes exponential time and is unrealistic for inputs of reasonable size). He initiates the research program of formalizing the notion of the “degree of difficulty of computing functions,” which we now call computational complexity.

General complexity measures. Rabin points out that time may not be the only resource we want to study; we may wish to consider other complexity measures that signify difficulty of computing functions. He thus proceeds to define complexity measures axiomatically.

Axiomatic, machine-independent approach. An important consideration when choosing axioms is machine independence. Even if we focus on a fixed resource, say time, the measured complexity of a problem depends on various details of the computational device, which would obscure rather than illuminate. Rabin therefore seeks conclusions that do not rest on one particular implementation. Rabin proposes a framework to achieve this, defining complexity measures using three axioms, and his hierarchy results (below) then hold for *every* measure satisfying these axioms.

Blum [21] later built on Rabin’s work, and developed a somewhat different, simpler, and more streamlined axiomatic framework for the same purpose, which became standard. Blum complexity measures are defined via two simple axioms, which are completely machine independent. Besides a hierarchy theorem for his measures, Blum additionally proved fundamental compression and speed-up results for them.

Decision problems. To avoid extraneous effects on complexity, such as the size of an output, Rabin focuses his theory on functions taking only the values 0 and 1, namely decision problems.

Asymptotic complexity. Rabin suggests that we should ignore the complexity of any finite set of inputs, and rather focus on eventual, limiting behavior of any algorithm. Fix a class of algorithms and a complexity measure m . We say that $f < g$ (read “ g is harder than f ”) if there is an algorithm A computing f such that, for every algorithm B computing g , one has

$$m_A(n) < m_B(n)$$

for all but finitely many n .

The partial order of complexity. For any admissible complexity measure, Rabin studies the partial order it induces on computable functions as defined above. He proves that this partial order contains infinite chains, and in particular functions of arbitrarily high difficulty for any such measure.

Theorem 5.2 (Rabin [183]). *For every computable function f , there is a computable function g taking only the values 0 and 1 such that $f < g$. If f is primitive recursive, g may also be chosen primitive recursive.*

For concrete resources, stronger and more detailed class-theoretic results became possible. Hartmanis and Stearns [103] developed explicit time-bounded classes, simulation results, and hierarchy theorems. A closely related follow-up by Hartmanis, Lewis, and Stearns [102] classified computations and proved hierarchy results for both time and space requirements. By now we have numerous such results.

We now turn back to focus on time complexity, and on the major choice of what the new theory of computational complexity should treat as an *efficient* or *tractable* algorithm.

5.2 The emergence of the class P

The choice of polynomial time algorithms as the theoretical counterpart of practically efficient ones was crucial for the field. Four influential early papers advocating this choice are those of Edmonds [64], Cobham [47], and Rabin [174, 185]. Their respective arguments and emphases are overlapping, and together make a strong case for this choice for the notion of *tractability*. We summarize their arguments below.

Cobham stresses the robustness of polynomially bounded computation across reasonable machine models (for example, various types of Turing machines). Cobham and Rabin emphasize that the closure of polynomials under addition and multiplication means that polynomial bounds on complexity are stable under common programming practices of composing computations. Edmonds and Rabin draw motivation from concrete algorithms: Edmonds from his algorithm for perfect matching in general graphs, and Rabin from his algorithms with Perles and Shamir [174] for the synthesis and analysis of automata. In both cases these algorithms have polynomial running times of low degree, while straightforward exhaustive procedures were exponential. In these early days before complexity considerations became widespread, Edmonds and Rabin felt compelled to explain why the former are much better than the latter, as exponential growth is much faster. Rabin adds a vivid way of illustrating it: if computers become faster by a constant factor, a fixed amount of time permits a multiplicative increase in input size for a polynomial-time algorithm, but only an additive increase for an exponential-time algorithm.

These arguments proved convincing, and the community adopted polynomial time as the central initial formalization of efficiency and tractability. This choice led directly to some of the most important complexity classes and questions. The class **P** contains problems with efficient deterministic algorithms; **BPP** contains problems with efficient probabilistic algorithms and bounded error; **NP** contains problems with efficiently verifiable certificates. Moreover, the **NP**-complete problems become the hardest problems in **NP**, under efficient reductions. These are only the most prominent examples: polynomial resource bounds enter the definitions of hundreds of classes and their relationships; see, for example, the Complexity Zoo [2].

The impact is broader still. Modern theories of cryptography and pseudorandomness replaced particular attacks and statistical tests with the class of *all* polynomial-time adversaries or distinguishers. The compositional closure properties of this class again allowed researchers to uncover structure and greatly develop these theories. These are specific examples of a more general principle, suggesting that basic beliefs pervading philosophy about knowledge had to radically change. The old (information theoretic) idea that you know everything which you can infer from your current knowledge was usefully and practically replaced in many settings by the new (complexity theoretic) idea that this holds only for what you can *efficiently* derive (e.g., in **P**) from your current knowledge (see the survey of Aaronson [1] for an excellent survey).

One cannot exaggerate the importance of this initial choice. In essence, the flexibility of polynomial time gave complexity theory *robust* notions with which to explore relationships among problems and models. These revealed many structures, such as complexity classes, as well as flexible techniques that created a working methodology of the field. As this understanding evolved, the theory could go on to study many more refined computational models and resources, leading to more refined results and questions, and to closer ties with practical efficiency and relevant technologies (as well as to beautiful mathematics).

5.3 Lower bounds

From its inception, computational complexity aimed to prove that some problems are difficult to compute. Rabin [183] opens his paper by asking whether inverting von Neumann’s function (which for an n -bit input x outputs the n “middle” bits of x^2) is hard. Cobham [47] starts (and ends) his paper by asking whether multiplication is harder than addition. Of course, we do not seem any closer to resolving the **P** versus **NP** problem than we were when it was formulated in the early 1970s.

Seven decades of attempts to prove hardness have taught us that this is among the hardest tasks in computational complexity. Progress along this front has been slow and has focused on restricted models of computation, with the hope of developing stronger and stronger techniques that

may allow progress. Rabin has been part of this effort, proving mainly lower bounds on arithmetic and algebraic circuit models, for example, in his papers [187, 188, 202, 18], which we do not review here. We conclude by describing a different important early contribution of Rabin to proving lower bounds.

In 1974, Fischer and Rabin [73] proved a powerful lower bound on the complexity of any decision procedure for Presburger arithmetic. Recall from Section 4 that this is the first-order logical theory of the integers, which allows addition but not multiplication. It was proved decidable by Presburger in 1929, but the complexity of his algorithm was nonelementary²⁸. How hard is it? Fischer and Rabin show that it requires at least doubly exponential time²⁹.

Theorem 5.3 (Fischer–Rabin [73]). *Any algorithm deciding the validity of formulas in Presburger arithmetic must run for at least 2^{2^n} time on formulas of size n .*

Their proof is a very sophisticated diagonal argument. To implement it, they encode the computation performed by any algorithm A on input x of length n as a formula in Presburger arithmetic, which is valid if and only if the algorithm halts in 2^{2^n} time. In essence, the formula of length n has to somehow encode the tableau of the entire computation, and check its correctness. Doing so for such a long computation—for example, accessing exponentially long addresses and extracting the bit stored at a given address—seems impossible in a very succinct formula with the severely restricted expressive power of this logic. Thus the heart of their argument is precisely inventing such an encoding.

To date, no better lower bound has been proved. Oppen [169] greatly improved the upper bound by giving a decision procedure that runs in triply exponential time. The single exponential gap remains an open problem.

6 Probabilistic algorithms

The field most associated with Rabin is that of probabilistic algorithms. Today this is a huge field, with numerous such algorithms, analysis tools, and application areas, which are the subject of many textbooks and standard courses. Rabin played a major and distinctive role in its rapid early development in the 1970s. A sequence of his papers (and lectures) modeled, exemplified, and enthusiastically advocated the use of randomness (and the toleration of error) in computation. They demonstrate how probabilistic algorithms can achieve better efficiency in various resources, as well as greater simplicity in their design and analysis, for many problems in diverse domains. In all of these, Rabin insisted from the start on the now-standard worst-case analysis over inputs, of the resources used and the error incurred by these algorithms.

This section describes some of his probabilistic models and algorithms, which in turn improve memory use, running time, communication, and simplicity over deterministic ones. In subsequent sections on cryptography and distributed computing, we will see more of his probabilistic algorithms and protocols, which outperform deterministic ones in different ways. We will also discuss Rabin’s work on modeling probabilistic computation and reasoning about it.

6.1 Background

Randomness has both mystified and been utilized by humans since ancient times. Many societies have used lotteries of different sorts for decision making and for gambling. Mathematical studies

²⁸Namely, requiring time growing faster than any fixed tower of exponentials.

²⁹Single-exponential time lower bounds had been given for some natural problems in the 1972 paper of Meyer and Stockmeyer [157], and also in the present paper of Fischer and Rabin for deciding the first-order theory of the reals.

of randomness started with probability theory (Pascal, Laplace), and proceeded through statistics (Galton, Pearson, Fisher), statistical mechanics (Boltzmann, Maxwell), algorithmic randomness (Kolmogorov, Solomonoff, Chaitin), information theory, and private-key cryptography (Shannon). The use of randomness as a novel mathematical tool arose in the probabilistic method (von Neumann, Shannon, Erdős).

The computing revolution brought the use of randomness into computation. Turing was perhaps the first to explicitly suggest using a “random element” in (learning) algorithms, in his famous paper on artificial intelligence [237]; he even suggested physical devices that would generate such randomness. Early uses of randomness in computation were primarily in simulations of natural phenomena, such as weather prediction and nuclear interactions, and in many other applications mentioned in the famous Metropolis–Ulam paper [155] expounding the *Monte Carlo* method. It was also used for numerical integration. These initial works were mostly for practical uses, without a formal model or performance analysis. More fundamentally, many of these applications were statistical in nature, where the need for randomness is almost explicit in the problem formulation, and for which deterministic algorithms seemed irrelevant. In contrast, most problems we shall consider in this section are discrete problems from different domains, such as decision or search problems. Classically, these were solved only by deterministic algorithms, and allowing randomness (which may have seemed irrelevant) offered surprising gains.

6.2 Modeling probabilistic algorithms

The first paper to suggest a formal probabilistic model of computation is by de Leeuw, Moore, Shannon, and Shapiro [51], who defined a probabilistic Turing machine in 1956. However, the authors study its power from a computability (rather than computational complexity) viewpoint. Their main result is that this model is not more powerful than the usual deterministic Turing machines; i.e., such machines can compute with high probability only decidable functions. The simulation is by the usual enumeration of all possible values of the random choices, and counting how many of them lead to acceptance. Thus, for decidability, there is no advantage in using probabilistic algorithms over deterministic ones.

The focus on computational complexity changes everything. The first paper to do so was written a few years later, in 1963, by Rabin [184]. It defines a different formal model, namely probabilistic finite automata. In the spirit of his paper with Scott discussed in Section 3, it continues to investigate finite space complexity. He proves, as with nondeterminism, that probabilistic algorithms can provably save space. As usual, Rabin’s paper provides a comprehensive treatment of the new model, its variants, properties, and their relationships. Here we describe the main import of this paper. In the next subsection we discuss his 1976 paper modeling and exemplifying time-efficient probabilistic algorithms.

Probabilistic finite automata are defined as a natural extension of deterministic and nondeterministic finite automata: for every current state and symbol read, the transition function gives a probability distribution on the next state. The machine accepts an input if the probability of reaching an accepting state exceeds a threshold λ .

From the outset, Rabin distinguishes two types of machines:

- “unbounded error,” in which an input is rejected when its acceptance probability is at most λ ; and
- “bounded error,” (or “gapped”) in which, for some *gap* $\delta > 0$, an input is rejected when its acceptance probability is at most $\lambda - \delta$, with a guarantee that no input has acceptance probability in $[\lambda - \delta, \lambda]$.

Analogously to the notation of Section 3, we denote a gapped one-way probabilistic finite automaton by 1PFA³⁰.

Following Rabin, this distinction was adopted, became central, and has since been investigated in complexity theory for a variety of computational resources; we mention some. For example, for probabilistic polynomial-time computations, we have the class PP of unbounded-error algorithms and the class BPP of bounded-error algorithms. In this model, we have several reasons to believe that the unbounded-error model is more powerful³¹. For other models and resources, like space complexity, communication complexity, and decision-tree depth, we can formally prove (without any assumptions) that the unbounded-error model is more powerful.

For finite automata, Rabin proved that this distinction is essential. On the one hand, he proves that the unbounded error model accepts undecidable languages (giving both a counting argument proof as well as an explicit one)³². On the other hand, he proves that the bounded error model computes only regular languages. Ever conscious of computational resources, Rabin also gives upper and lower bounds on the blow-up in the number of states (the memory requirement) in simulating probabilistic machines by deterministic ones.

Theorem 6.1 (Rabin [184]).

- $1PFA = 1DFA = REG$.
- An n -state, δ -gapped 1PFA can be simulated by a $(1/\delta)^{O(n)}$ -state 1DFA.
- For every δ there exists a δ -gapped two-state 1PFA such that any 1DFA simulating it requires $1/\delta$ states.

In the proof of the second item, the states of the 1DFA simulating a given 1PFA are an appropriate $\epsilon(\delta)$ -net of probability distributions over the states of the 1PFA, with the natural transition function between them. This may be viewed as a quantitative generalization of the famous subset construction in the Rabin–Scott simulation of 1NFAs by 1DFAs.

Rabin proceeds to study a natural computational question suggested by this simulation, namely approximating the product of a given sequence of stochastic matrices. He points out that the naive approach cannot be space-efficient, even if the entries are small rationals, because of the numerical precision needed as the sequence grows longer. He then proposes a natural condition on the inputs under which a direct reduction to the simulation above actually works in finite memory. This basic problem was later studied in detail, as it was naturally recognized as a complete problem for simulating probabilistic log-space machines; see, for example, [26, 212, 112] for the history and best-known results. The paper further studies how other aspects of a model as simple as finite automata are affected by the injection of randomness, including the stability of probabilistic automata under perturbations of transition probabilities, as well as special types of such automata; we shall not elaborate on these.

Instead, we conclude with later work on variants of this model, which further illustrates its subtleties. Curiously, Rabin does not study 2PFAs, namely bounded-error probabilistic automata that, as in deterministic automata, allow two-way reading access to the input. Their power was discovered by Freivalds [76], who showed that they recognize the nonregular (but context-free)

³⁰Rabin calls the threshold λ a *cut-point*, and the bounded-error model one with an *isolated cut-point*.

³¹On the one hand, PP contains NP, which is believed to require exponential-size circuits. On the other hand, BPP is known to have polynomial-size circuits. A separation of PP and BPP also follows if the polynomial-time hierarchy does not collapse, combining [232] and [223, 138].

³²This result necessitates arbitrary real numbers as threshold. If one demands rational (or even a computable real) threshold, and rational transition probabilities, the resulting language is decidable.

language $\{0^n 1^n : n \geq 1\}$. This is surprising as it informally means that, with high probability, *fixed* constant space suffices to count to n for any *unbounded* integer n .

This clever algorithm (which the reader is challenged to discover) runs in exponential time in the input length. Perhaps even more surprisingly, Dwork and Stockmeyer [61] proved that, when restricted to polynomial expected time, 2PFAs can recognize only regular languages. Models that combine randomness and nondeterminism in finite automata were studied as well, and their exact power is still a mystery (see [48] for partial progress and an intriguing conjecture).

6.3 Primality testing: randomness for algebra and number theory

In April 1976 Joe Traub organized a meeting at Carnegie Mellon University titled “Algorithms and Complexity: New Directions and Recent Results.” The invited speakers included many leaders of the already quite diverse field at the time (including Berlekamp, Brent, Garey, Johnson, Karp, Knuth, Kung, Paterson, Rabin, Shamos, Simon, Tarjan, Traub, and Yao). The proceedings of the meeting [234] present quite a panorama justifying the title: the volume displayed the remarkable breadth of the exciting new trends and directions then emerging. Rabin was there to excite the audience about probabilistic algorithms. Excitement was needed: Rabin felt compelled to state explicitly in his paper, “It may at first seem surprising that employing randomization leads to efficient algorithms,” a statement no one would dream of uttering a decade later. Rabin’s paper, “Probabilistic Algorithms” [189], is the focus of this subsection, and the next one. But first, some background.

The 1970s saw probabilistic algorithms become a central algorithmic tool to be considered for any problem that deterministic algorithms were used to solve, in the hope of beating them! Some famous examples existed even beforehand. A unique occurrence, long before computers were dreamed of, is Pocklington’s 1917 paper [175] on computing square and cube roots in finite prime fields. While lacking precise language and a formal model, the paper clearly gives a polynomial-time probabilistic algorithm for these problems (which to this very day do not have a deterministic counterpart without number theoretic assumptions!). This paper remained generally unknown, but the idea was rediscovered several times (for example, by Peralta, Lehmer, and Berlekamp [173, 141, 20], as we shall see below). Another famous example is Hoare’s 1961 paper [108, 109] introducing his probabilistic sorting algorithm *Quicksort*. While it has no asymptotic advantage over the $O(n \log n)$ deterministic sorting algorithms such as *mergesort*, Hoare argues that his algorithm is simpler, easier to implement, and faster by a constant factor in expectation.

An important paper in the development of probabilistic algorithms is Berlekamp’s 1970 paper [20] on factoring univariate polynomials over large prime fields. Polynomial factoring is a central, basic algebraic problem. It generalizes root taking (above) and plays an important role in decoding algebraic codes such as Reed–Solomon codes. Importantly, we do not know any deterministic polynomial-time algorithm for this problem.

Theorem 6.2 (Berlekamp [20]). *There is a probabilistic expected-polynomial-time algorithm that, given a polynomial $f(x)$ and a prime p , produces the (unique) factors of f in $\mathbb{F}_p[x]$.*

But perhaps the largest impact demonstrating the algorithmic power of randomness came from the probabilistic *primality-testing* algorithms of Solovay–Strassen [225] and Rabin [189, 191] in the late 1970s.

Theorem 6.3 (Solovay–Strassen and Rabin [225, 189, 191]). *There is a polynomial-time probabilistic algorithm for testing whether a given integer is prime or composite. It is always correct on prime inputs and errs with low probability on composite inputs.*

It may be important to point out a source of power of randomness in such algebraic and number-theoretic settings, which may differ from that in other applications. Without describing any of the algorithms in the theorems above, let me note how algebraic “rigidity” easily supplies the error bounds for them. Essentially, random choices are bad (lead to error) only if they fall into a “small” algebraic substructure of the structure they are chosen from (for example, a proper subgroup or coset, or an algebraic subvariety of a vector space). In the polynomial-factoring algorithms and the primality testing algorithms, the random choices are typically elements of a group G (for example, $\mathbb{Z}_N^*$, the multiplicative group of invertible elements modulo N for some integer N). Simple analysis shows that a random choice leads to an error if and only if it lies in a proper subgroup H of G (e.g., containing elements of a trivial multiplicative character value). Then, Lagrange’s elementary theorem that the size of a subgroup always divides the size of the group directly shows that the error has probability at most $1/2$! ³³

The impact of these two (independent but very similar) primality tests was huge. They announced the arrival of probabilistic algorithms as an independent field, causing everyone seeking algorithms to consider employing randomness just as naturally as seeking a deterministic algorithm. Part of the appeal and inspiration of the probabilistic primality tests must have come from the fame of the problem itself (for example, even Gauss appealed to the mathematics community to find efficient ways to test primality³⁴). Another attraction must have come from how essential large primes became to public-key cryptography with the foundational works of Diffie–Hellman, Rivest–Shamir–Adleman [55, 206], Goldwasser–Micali [91, 92], and the wave of theory and practice that followed.

Indeed, the period of the late 1970s and the 1980s saw an explosion of probabilistic algorithms in numerous domains. These included polynomial identity testing [54, 214, 247], universal hashing [34], randomized routing [239], parallel matching algorithms [122, 164], randomized rounding [203], and many others.

Let us return to Rabin’s 1976 paper [189], as it contains far more than the primality-testing algorithm. Rabin clearly set out to excite that conference audience, by laying a foundation for a new and exciting subfield of algorithms, and arguing for its importance and potential with new examples and open questions. Again, today many of these are considered standard and obvious, but apparently this was not so at the time.

Rabin begins his paper by distinguishing these algorithms from another subfield, probabilistic analysis of algorithms (the subject of Karp’s lecture at the same conference), in which one analyzes *average-case* behavior of deterministic algorithms under some natural input distributions. Rabin stresses that probabilistic algorithms demand *worst-case* bounds on their complexity and error bounds. He lays out the model in some detail. A complete formalization and the familiar classes followed a year later in Gill’s paper [83]. He discusses the variants that we today call BPP, RP, ZPP: respectively, two-sided-error, one-sided-error, and zero-error models. Only then does Rabin demonstrate the power of randomization by giving two novel efficient probabilistic algorithms from very different domains: number theory (the primality test) and geometry (which we discuss in the next section). These two algorithms use very different techniques.

³³Another important set of examples of how algebraic structure is utilized is when the random choices are taken from a vector space, and lead to error when a low-degree polynomial vanishes on them. For univariate polynomials (a vector space of dimension 1, namely one variable), the degree of the polynomial trivially upper-bounds the number of roots (on which it vanishes). This leads to a simple error bound when the number of possible choices is larger than the degree. For multivariate polynomials the same error bound holds as well, due to the important but simple-to-prove multivariate extension in [54, 214, 247] of the univariate case. This principle underlies many important probabilistic algorithms, including the celebrated algorithm for *multivariate* polynomial factoring over finite fields due to Kaltofen [116, 117] (here both input and outputs are given by straight line programs).

³⁴And to factor integers.

We proceed with the primality story we started earlier. Rabin notes that his algorithm is really a version of a deterministic primality test of Miller [158],³⁵ which runs in polynomial time under the ERH (the Extended Riemann Hypothesis, a generally believed conjecture in number theory; it guarantees that good random choices exist in a small interval, which may be searched deterministically). Rabin suggested that ERH may be false, and that a random search will find good random choices with high probability. Another point Rabin makes is that his (and Solovay–Strassen’s) one-sided algorithm certifies composites with high probability but does not certify primes. Given that short certificates for primes do exist (as Pratt showed in [177]), Rabin asks for a one-sided algorithm in the other direction, which would find such certificates with high probability. Such algorithms were eventually found, first by Goldwasser–Kilian [90] (under mild number-theoretic assumptions) and then by Adleman–Huang [6] (without them). These algorithms and their analyses are much subtler and surprisingly use the theory of elliptic (and higher genus) curves. Combined with the original algorithms, one gets zero-error primality tests running in expected polynomial time, which always certify their answer. The beautiful ending of this story is, of course, the deterministic polynomial-time algorithm for primality by Agrawal, Kayal, and Saxena [8]. It is worth mentioning that it emerged from ingeniously *derandomizing* yet another probabilistic primality test of Agrawal and Biswas [7], which is very different from those above. The best version of this deterministic algorithm runs in about $O(n^5)$ time, and to this day most practical systems (for example, in cryptography) use the much faster, roughly $O(n^2)$ -time original probabilistic primality tests!

Rabin continued looking for algebraic and number-theoretic problems to exemplify the power of randomness. In joint work with Shallit [201], he gives a probabilistic algorithm for finding a representation of a given integer as a sum of four squares; the existence of such representations is guaranteed by a famous theorem of Lagrange [136]. In [192], Rabin gives probabilistic algorithms for several problems in finite fields, including finding irreducible polynomials, finding roots of univariate polynomials, and factoring them.

6.4 Closest pairs: randomness for geometry

Another subfield of algorithms that burst into existence in the 1970s was computational geometry. Among several initial works, a major impetus came from Shamos’s Ph.D. thesis [218], papers, and his survey of 75 problems [217]. Basic problems in which the typical input was a collection of points in Euclidean space were addressed, many arising from practical questions about the real world. These problems differ from the general combinatorial and graph-theoretic questions on which the algorithmic community was focused, as geometry imposes extra structure, including convexity, order, and metric structure. Thus, completely new questions could be asked, for example, about convex hulls and Voronoi diagrams, while old questions such as shortest paths and minimum spanning trees in weighted graphs were restricted to the special case in which weights came from Euclidean distances between points. New techniques were developed to design and analyze algorithms for such problems.

To demonstrate the power of randomness in this setting, Rabin picks the *closest-pair problem*: given a set of n points in the Euclidean plane, find the pair that is closest. The best deterministic algorithm for this problem, due to Hoey and Shamos [219], runs in $O(n \log n)$ time³⁶. Moreover, like many geometric algorithms, it relies on sorting, and so this running time seems the best one can hope for. Rabin shows that with randomness it can be solved in linear, $O(n)$ time.

Theorem 6.4 (Rabin [189]). *There is an expected $O(n)$ -time probabilistic algorithm that finds the*

³⁵Indeed, it is often referred to as the Miller–Rabin test.

³⁶This algorithm actually solves a more general problem, of finding a closest point for every point in the set.

closest pair among n points in the plane. The same holds in $\mathbb{R}^d$ for any fixed dimension d .

Unlike primality testing, where both the algorithm and its analysis are simple and require a few lines each to describe, here both are considerably more complex and subtle³⁷. With this example Rabin shows that the use of randomness can be far richer than one-shot sampling that is guaranteed to land in a large set. Let me briefly sketch the main ideas behind his solution.

The algorithm is recursive and demonstrates the power of subsampling. Given a set $P = P_0$ of n points, we select a subset P_1 of $n/2$ of them at random, then a subset P_2 of these of size $n/4$, etc. The idea is to use the solution to P_i to solve the problem on P_{i-1} in linear time. This leads to a time recurrence of the form $t(n) = t(n/2) + O(n)$, yielding $t(n) = O(n)$.

To facilitate the inductive step, Rabin proves that a strong invariant is maintained with high probability. If the shortest distance between a pair of points in P_i is, say $2\delta_i$, then a grid of this resolution (specifically of side length δ_i) will have at most one point of P_i (and, in particular, the closest pair will be in neighboring squares in this grid). Now add the points of $P_{i-1} \setminus P_i$ back to this grid. Then solve the closest-pair problem in the neighborhood of each nonempty cell by brute force, comparing all pairs in it. This is a quadratic amount of work per cell, and so the crux of the analysis is proving that the randomness of these sets guarantees, with high probability, that the sum of these squared occupancies is still $O(n)$.

Needless to say, a full description of the algorithm and the necessary data structures to support it efficiently is considerably more complex (indeed, some of these details are missing, and were supplied later: see the paper of Banyassady and Mulzer [16] for full details and simpler analysis).

Subsampling and recursive subsampling became basic tools in probabilistic algorithms. One beautiful example is the linear-time probabilistic algorithm for minimum spanning trees of Karger, Klein, and Tarjan [119], which has a very similar high-level structure.

6.5 Fingerprinting: randomness and communication

Another powerful demonstration of randomness gives provably exponential savings in communication. The basic setup, laid out in Yao’s seminal paper [243] on communication complexity, is the following. Two players, Alice and Bob, hold n -bit strings x and y , respectively, and must compute a fixed function $f(x, y)$ using a fixed protocol of their choosing. The communication complexity of f is the minimum number of bits they must exchange to determine the function value for every input.

Easily the most basic function to consider is equality: testing whether $x = y$. In lectures and classes, Rabin motivated this problem, as well as the focus on communication costs and ignoring computational ones, through what he called the “man on the moon” problem. Alice on Earth and Bob on the moon want to verify consistency of the two huge data files they hold. Of course, one of them can send their whole input to the other, at a possibly prohibitive cost of n communicated bits. This unfortunately is best possible; it is not hard to see that in any deterministic protocol they must exchange at least n bits. The Rabin–Yao probabilistic protocol³⁸ does exponentially better!

Theorem 6.5 (Yao [243]). *There is an $O(\log n)$ -communication probabilistic protocol for equality with inverse-polynomial error.*

The protocol and its analysis are extremely simple. Alice picks a small random prime $p \leq n^2$, and sends Bob both p and $x' = x \pmod{p}$, only $O(\log n)$ bits in total. Bob computes $y' = y$

³⁷Even when I oversimplify and ignore the precise machine model.

³⁸It is referred to by Yao [243] as joint unpublished work with Rabin in the paper above, and is attributed to both in all standard texts [162, 134, 159].

(mod p) and decides according to whether $x' = y'$. Clearly, if $x = y$, this decision is always correct. If $x \neq y$, at most n distinct primes³⁹ can divide $x - y$; thus, the error probability is at most $O(1/n)$.

The very same idea was independently discovered and used at about the same time for a related, though different, demonstration of the power of randomization for one-tape Turing machines. It was known [17, 104] that such machines require $\Omega(n^2)$ time to recognize the equality language, namely input strings of the form xx . Freivalds [75] uses the same reduction modulo a small random prime to show that randomness gives a near-quadratic saving.

Theorem 6.6 (Freivalds [75]). *There is a probabilistic one-tape Turing machine that recognizes the equality language with high probability in time $O(n \log n)$.*

This idea of *random fingerprinting* is fundamental and in universal use! Indeed, it is so basic that it is regularly taught in undergraduate classes without giving a reference for its origin. Needless to say, people used reduction modulo primes as simple *hash*⁴⁰ functions in practice, but typically used fixed primes. Freivalds, Rabin, and Yao all stress that random fingerprinting works for *every* input with high probability; despite compressing large objects into tiny fingerprints, collisions rarely occur!

Rabin kept seeking different methods and different applications of random fingerprinting. In his paper [193], Rabin suggested that, instead of viewing bit strings as encoding integers and reducing them modulo primes (which requires modular arithmetic), one can view them as $\mathbb{F}_2$ -polynomials and reduce them modulo irreducible low-degree polynomials (which requires only simple local bit operations). A different method, encoding bit strings by random $\text{SL}_2(p)$ matrices, is proposed and analyzed in his paper with Karp [121], which is the subject of the next subsection.

6.6 Pattern matching: randomness and simplicity

Rabin’s next demonstration of the algorithmic power of randomness concerns a central problem that already had an optimal algorithm. Here the power is to *simplify*: to replace complex algorithms by a simpler one with a simple analysis, broad generalizability, and better practical efficiency.

Pattern matching is the following basic problem in text processing. Given two sequences (say, binary, although any alphabet will work), a pattern x (of length m) and a text y (of length $n \geq m$), the goal is to find occurrences of x as a consecutive block in y .

A trivial $O(mn)$ algorithm suggests itself. However, Knuth–Morris–Pratt [130] and Boyer–Moore [27] found ingenious $O(n)$ -time algorithms⁴¹ for this problem. The Karp–Rabin paper *Efficient Randomized Pattern-Matching Algorithms* [121] applied the random fingerprinting method above to give a probabilistic algorithm that runs in expected $O(n)$ time. Despite not improving the time complexity, they make the important point that simplicity (in exposition, analysis and/or implementation) may sometimes justify the use of randomness and the tolerance of error. Moreover, their algorithm is far more space-efficient than the deterministic one.

Theorem 6.7 (Karp–Rabin [121]). *There is an $O(n)$ -time, $O(\log n)$ -space probabilistic algorithm for pattern matching with negligible error.*

Again, the idea is strikingly simple. As pattern matching calls for a sequence of equality tests (of x to length- m “windows” of y), fingerprinting is natural to use. Thus, we fingerprint x by a logarithmic-size value x' as above, similarly fingerprint the first m -window of y by y'_1 , and compare

³⁹Indeed, at most $n/\log n$ primes, if one uses standard results on the distribution of primes.

⁴⁰A synonym for fingerprint.

⁴¹All algorithms in this subsection are in the RAM model, which allows constant-time operations on logarithmic-size words.

the fingerprints. The main observation is that, as the window slides one symbol to the right, updating the fingerprint y'_i to y'_{i+1} requires only a *constant* number of additions, subtractions, and comparisons, resulting in a linear-time algorithm⁴².

Note that in this algorithm (and indeed, the deterministic ones), the text y can be given as a stream rather than being presented fully before the computation begins. Unlike the deterministic algorithms, which require memory $\Omega(m)$ bits, the probabilistic one uses only $O(\log n)$ memory. Streaming algorithms with low memory requirements are by now a huge field (see, for example, [165]), and the Karp–Rabin algorithm can be seen as a very early example. For more details on this viewpoint see [29].

6.7 Logic for probabilistic computation

Rabin’s interest in randomness was not limited to algorithm design. Once probabilistic computation becomes a central paradigm, one also needs formal systems for reasoning about it.

In joint work with Halpern, Rabin studied logics for reasoning about likelihood and probabilistic computation [96]. This work belongs conceptually with the logical themes of the earlier sections, but it is naturally discussed here because the computations being reasoned about are probabilistic.

The point is that probabilistic computation introduces new kinds of assertions. One may wish to say not only that an event occurs, but that it occurs with high probability, or that one event is more likely than another. Such statements require logical languages richer than ordinary Boolean assertions.

For this purpose, Halpern and Rabin introduce a new qualitative modal logic LL for probabilistic computation, and compare it to other proposals for that purpose. For LL , they gave possible-worlds semantics, proved a complete axiomatization, and gave an exponential-time decision procedure for satisfiability.

This completes a circle in Rabin’s work. In the decidability section we saw logic used to reason about computation and automata. Here, after randomness has become an algorithmic tool, logic is again brought in to reason about the new computational phenomenon. Further information about subsequent works and this general field can be found in the monograph [95] and the survey [53].

7 Cryptography

Cryptography is another vast subject; again, we introduce it only briefly, and we refer the reader to the textbooks of Goldreich [85, 86] and of Katz and Lindell [123], or the much shorter cryptography section in [241].

Since time immemorial, people have had secrets that they wished to share with some and keep private from others. Thus, schemes for secret communication were developed in very ancient cultures. What was common to all of them, from Caesar’s substitution code to the Enigma machine and everything in between, was the assumption that the parties wishing to communicate secretly already shared some secret information. Moreover, this assumption was formally justified by Shannon, who proved, using his newly developed *information theory*, that the parties’ shared secret key must be completely random and at least as long as the message they send if they want no information about it to be leaked to an eavesdropper [220]⁴³.

⁴²If we want to find, for example, the first occurrence of x , we can even make the algorithm zero-error and still run in expected linear time by checking the putative match (investing time m whenever $x' = y'_i$), as the probability of a false match can be reduced below $1/(mn)$ by picking a random prime below, say, $(mn)^2$.

⁴³The alert reader must have noticed the circularity here: to communicate secretly at some point, one must have communicated secretly at some earlier point, and so on. Shannon proves that this cannot be avoided if perfect

The revolution that completely removed the need for this “essential” assumption came in 1976 with the breakthrough paper of Diffie and Hellman [55], which introduced computational complexity into cryptography. Their paper proposed making the entirely reasonable assumption that all participants are computationally limited (to polynomial time) and, furthermore, that some problems cannot be solved so quickly. In particular, they proposed the *discrete logarithm* problem as such a hard problem. Under this assumption, they showed that secret communication is possible between any two parties sharing no prior secrets whatsoever. To understand how this circumvents Shannon’s barrier, we must understand a fundamental difference between the information-theoretic and complexity-theoretic settings. In the former, every representation of a piece of information is equivalent to every other. In the latter, this symmetry can be broken, as some representations are easier to compute than others.

Fix a prime p and a generator g of the multiplicative group $\mathbb{Z}_p^*$. Then consider the (bijective) exponentiation map

$$x \mapsto y = g^x \pmod{p}.$$

Obtaining y from x is easy, using repeated squaring of g modulo p , requiring only polynomial time. However, obtaining x from y is the discrete logarithm problem, which is assumed intractable. Thus, in a computationally limited world, these two representations are very different. Diffie and Hellman exploit this asymmetry to create *public-key cryptography*, in which any two parties sharing no prior secret information can efficiently exchange secret messages that cannot be decoded by any other computationally bounded party. They also give a completely new cryptographic application: personal digital signatures that others cannot forge.

The main impact of the Diffie–Hellman paper was suggesting a new paradigm for cryptographic security: to prove that your protocol (for any particular task) is secure, prove that breaking it will result in an efficient algorithm for a clean, hopefully well-studied mathematical problem. One such problem, perhaps *the* problem, was integer factorization, which has been studied for millennia (in particular, Gauss famously appealed to the mathematical community to find a fast algorithm for it). And it seems hard: the best known classical factoring algorithm is by Buhler, Lenstra and Pomerance [32]; it requires time about $\exp(n^{1/3})$ to factor n -bit integers⁴⁴.

The hardness of factoring soon became the assumption of choice in theory and practice. The first to introduce it were two papers, by Rivest-Shamir-Adleman and by Rabin. Rivest, Shamir, and Adleman [206] introduced their famous RSA public-key cryptosystem, whose security is based on a problem related to the complexity of factoring integers, while Rabin introduced a public-key cryptosystem whose security is *equivalent* to the complexity of factoring [190] (both systems provide digital signatures as well). These works still fell short of being able to securely encrypt single bits or to argue about partial leakage of information. The seminal 1982 paper of Goldwasser and Micali [91, 92] introduced *probabilistic encryption* and laid formal mathematical foundations for the field, giving precise definitions of security and a public-key system that attains them based on the hardness of another problem closely related to integer factoring.

The decade following the Diffie–Hellman paper saw a frenzy of activity in the newly formed community of modern, complexity-based cryptography, exploring which other apparently impossible tasks were possible in this new wonderland. It was a period in which one could “ask for the moon” and not be entirely surprised to get it. Thus were born protocols for such cryptographic primitives and goals as *coin flipping into a well*, *poker playing over the telephone*, *zero-knowledge proofs*, and

information-theoretic secrecy is required.

⁴⁴In 1994 Shor [222] discovered a poly-time *quantum* algorithm, so if quantum computers become a reality the hardness of factoring can no longer serve cryptography; indeed, practical security systems today are being replaced by other, hopefully “quantum resilient” assumptions.

many other feats. Together with new techniques for analyzing protocols, and new models and assumptions, the road was paved for the Internet revolution that followed in their wake.

In the next two subsections we discuss two important contributions of Rabin to this activity. We note in passing that Rabin, whose primary passion was randomness and its utility, was naturally drawn to cryptography, a field in which randomness is essential.

7.1 Oblivious transfer and secret exchange

Of the many ingenious cryptographic primitives invented in this hectic period, one of the most surprising and, as we shall see, fundamental is Rabin’s *oblivious transfer* [194]. We have two parties, conventionally called Alice and Bob. Alice holds a secret. An oblivious transfer is a protocol in which Alice and Bob participate, achieving two goals: first, Bob receives Alice’s secret with probability exactly one-half; and second, Alice learns nothing about whether Bob received it or not.

My first reaction—and probably that of many others—on hearing this for the first time was, “Why?” Who in the world would want such a functionality? In what situations would Alice prefer *not* to know whether her secret went through?

Rabin, of course, knew the answer. He first constructed a simple oblivious-transfer protocol based on the hardness of factoring. He then applied this seemingly useless primitive to design a protocol for one of the challenging tasks at the time: *secret exchange*. In this task, Alice and Bob each have a secret, and we seek a protocol (with extremely high probability) ensuring that Alice gets Bob’s secret *if and only if* Bob gets Alice’s secret⁴⁵.

Theorem 7.1 (Rabin [194]). *Oblivious transfer implies secret exchange.*

This demonstration of the power of oblivious transfer was the first in a long sequence. Soon after Rabin’s paper appeared, Even, Goldreich, and Lempel defined a variant they called “1-out-of-2 oblivious transfer” (which is equivalent in power to Rabin’s) and showed further applications, some of which we shall see below [68]. A full understanding of that power of OT came later, when Kilian [124] proved that oblivious transfer is complete for *secure multiparty computation*.

Theorem 7.2 (Kilian [124]). *Oblivious transfer implies secure multiparty computation.*

Secure multiparty computation is, in a sense, the mother of all cryptographic protocols. It allows a set of mutually distrustful parties to evaluate jointly *any* efficiently computable, possibly randomized function f of their secrets. Indeed, being able to do this allows one to solve a vast class of cryptographic tasks. Yao [244, 245] defined this problem and solved it for the two-party case. His protocol starts with a Boolean circuit for f . One party first “garbles” the circuit and then the second party evaluates that garbled circuit. Both are carried out in a gate-by-gate fashion, according to the order of the circuit computation, where the garbling of each gate uses the oblivious transfer primitive. Goldreich, Micali, and Wigderson extended Yao’s work to secure multiparty computation by any number of players [88]. Their protocol is different; all parties jointly evaluate successive shared encodings of intermediate gate values (again using OT), which are unintelligible to any individual party. These results assume an underlying public-key encryption.

Oblivious transfer is a special case of secure multiparty computation, which seems much easier. Kilian’s theorem gives a converse: oblivious transfer is sufficient for general secure multiparty

⁴⁵Rabin makes an assumption of a physical nature, which was later removed in a sequence of works starting with Blum [22] on what became known as “fair secret exchange” using “gradual disclosure,” in which OT played a central role as well.

computation (without assuming public-key encryption), making Rabin’s apparently strange primitive one of the central building blocks of cryptographic protocol design ⁴⁶.

7.2 Public randomness and beacons

Complexity-based cryptography is partly the art of inventing assumptions that allow one to implement a variety of cryptographic primitives and applications. Even at the inception of the field, a distinction was made between primitives requiring only *one-way functions* and those requiring the stronger assumption of *trapdoor functions*⁴⁷. Since then, numerous finer or very different assumptions have been made to satisfy newly required functionalities. Needless to say, making the weakest possible assumptions is desirable, and a whole subfield of cryptography is devoted to studying their relationships.

A striking early contribution to this art was Rabin’s proposal for a very different type of assumption: having *public randomness* as a cryptographic resource. In his work on transaction protection by beacons [197], Rabin suggested a public service that broadcasts fresh random values to all participants. This assumes trust in a central entity, but the trust is minimal: it is limited to the supply of bits, independently of their use in any protocol. The idea was that a common source of unpredictable public randomness could simplify some cryptographic protocols and enable some that are impossible otherwise. In his paper, Rabin devised protocols in this setting for three related but different problems arising in digital communication, which are easy to implement when people meet in person or use trusted parties:

Contract signing. Alice and Bob each have a copy of a contract C . A protocol must ensure that Alice receives Bob’s signature on C if and only if Bob receives Alice’s signature on C ⁴⁸.

Confidential disclosure. Alice holds a secret D , which she is willing to share with Bob on the condition that he not divulge it. A protocol must ensure that Bob receives D if and only if Alice receives a legally binding statement from Bob that he will keep it secret.

Certified mail. As with paper mail, Alice wants to send a message M to Bob and have its receipt certified. A protocol must ensure that if Bob receives M , then Alice obtains a certificate that Bob received it.

This idea has become increasingly important. Public randomness appears in numerous protocols for coin flipping, electronic voting, noninteractive proof systems, distributed protocols, and modern blockchain systems; see, for example, direct continuations of Rabin’s beacon proposal and later public-randomness services [163, 115, 139, 70, 228]. In different applications, the assumed properties of the available public randomness vary—even Rabin’s paper uses two different forms—but the essential structure remains that all parties have access to this resource. In some applications, public randomness has been proved essential⁴⁹, whereas in others, it has later been shown to be removable.

⁴⁶We note that the relative power of the primitives oblivious transfer and public-key encryption is not known, but there is no way to get one from the other via “black-box” reductions [82].

⁴⁷For example, it took a long time before *digital signatures*, originally based on trapdoor functions, could be implemented assuming only one-way functions [207]. Likewise, the powerful *zero-knowledge proofs* require only one-way functions [89].

⁴⁸This looks superficially similar to the *secret exchange* problem above because of the simultaneity requirement, but the transferable legal evidence makes the two functionalities different.

⁴⁹Often under assumptions such as random oracles.

The latter demonstrates the power of the paradigm as much as the former: it allows one to solve a problem gradually, first using public randomness and then removing it⁵⁰.

Rabin’s three applications of public randomness above provide one example of the latter. In the aforementioned paper, Even, Goldreich, and Lempel employ their version of Rabin’s oblivious transfer to remove the need for public randomness in these applications [68].

Our final substantive subsection demonstrates a completely different application of public randomness.

7.3 Bounded storage and everlasting security

In an insightful paper, Maurer [149] suggested replacing the standard assumptions in complexity-based cryptography that limit the time available to participants in cryptographic protocols with an assumption limiting their storage space. His *bounded-storage* model further assumes public randomness, conceptually similar to Rabin’s *beacon* [197]. Specifically, we assume a stream of random bits, available to all participants, that is much longer than the given space bound. This paper initiated a beautiful line of work that achieves information-theoretic security for some cryptographic tasks under these natural assumptions. We informally describe the evolution of ideas and results, in which Rabin played a major role.

Maurer considers secret communication. Recall that, for information-theoretic security, Shannon already proved that if Alice wants to send Bob an n -bit message that looks completely random to an eavesdropper Eve, then Alice and Bob must share in advance a completely random secret key of length n . Maurer shows that, in his model, they can share far shorter keys and still obtain complete information-theoretic security, except on a vanishing-probability event. Thus, this model provides a way to overcome Shannon’s limit. Maurer also makes the point that this security holds even if the original shared secret key is later revealed. However, Maurer makes a strong restrictive assumption about the way Eve can use her memory: she can store only a subset of the bits from the public randomness. He raises the question of removing this assumption: can Eve process the public randomness as it is streamed and store any bounded amount of information about it?

In a sequence of papers with his former and current students [13, 12, 57], Rabin removed this assumption and did considerably more. In particular, they:

- proved that Eve may store arbitrary information about the public randomness, provided her storage capacity is sufficiently smaller than the length of that randomness;
- proved that Alice and Bob can use the same key for exponentially many message exchanges, achieving the same information-theoretic security as Maurer;
- extended the applications beyond secret communication to authentication and non-malleability; and
- formalized *everlasting security*, which Maurer noted above. They proved that their protocols guarantee that after the protocol is over, even if Eve obtains the shared secret key used by Alice and Bob and her space bound is lifted (namely, she has unbounded storage), the messages sent retain the same information-theoretic security.

The papers above did not achieve optimal parameters. Near-optimal parameters were obtained later by Dziembowski and Maurer [62]. A more general understanding of this model came with

⁵⁰The same has happened many times with randomized algorithms: a probabilistic algorithm is introduced first, and later a derandomization makes it deterministic. Indeed, as we mentioned in Section 6 the famous AKS deterministic primality test [8] is an extremely clever *derandomization* of the probabilistic Agrawal–Biswas test [7].

Lu’s paper [144], connecting the bounded-storage model with *randomness extractors*. This was followed by a paper of Vadhan [238], which constructed appropriate *locally computable* extractors and allowed most existing protocols and results to be generalized, simplified, and given improved parameters.

8 Distributed computing

Distributed computing captures any situation in which many computers or processes aim to achieve a variety of goals (common or separate) in a way that necessitates interaction. This, of course, encompasses numerous situations in computer science and technology (as well as in the natural sciences), making distributed computing a vast and heterogeneous field. General overviews of this field can be found, for example, in the books of Lynch [146] and Attiya and Welch [11], or the much shorter section on distributed computing in [241]. We note that distributed computing is similar to cryptography in many ways, as both involve many interacting parties. However, the issues central to each are typically quite distinct, as we shall see⁵¹.

One focus of many distributed systems and problems is *locality*. A distributed protocol assigns to every computer in the system its own program, and there is no central entity controlling them all. Each such local program iterates steps (or rounds) of internal computation (usually viewed as free) and communication exchanges with its neighbors. The latter can take place through various media: shared variables, communication channels in a given network topology (which is often not known to the individual processes), and so forth.

Besides locality, distributed protocols have to defend against faults. These can come in a variety of forms. Obvious examples include failures of, or even adversarial behavior by, some processors or communication media. Another difficulty, which will appear in some of the examples below, arises from *asynchrony*. In synchronous systems, we assume a global clock, and so the rounds of internal computation and message exchange with neighbors are synchronized. But in many realistic situations one cannot assume a global clock, and the local clocks may bear no relation to one another. This asynchrony is typically modeled by an adversary that has a global view of the system and can use it to schedule processor actions and message delays arbitrarily. Even in synchronous settings, we let such an all-knowing adversary dictate which processes fail and how, often dynamically during execution. A distributed protocol has to achieve its goals typically against such powerful adversaries (whose precise nature may differ and is specified for each problem). This adversarial power explains the difficulty of designing distributed protocols and the subtleties involved in proving their properties⁵².

In this section we survey major contributions made by Rabin to four fundamental problems (of very different natures) in distributed computation: *dining philosophers*, *mutual exclusion*, *Byzantine generals*, and *information dispersal*. The issues they raise are as relevant to practical applications today as they were when formulated decades ago. One need only consider the Internet and modern distributed-ledger protocols to see this.

A common element of the first three problems is the demand for *symmetric* protocols, namely, protocols in which all processors run identical local programs. In all three, we will see the power of randomness in action in distributed protocols, which allows them, in different ways, to break this symmetry. We note that such use of randomness was a key element in practical distributed systems,

⁵¹Naturally, there is a nontrivial intersection and plenty of interaction between these fields: cryptography is used in many distributed algorithms, and distributed algorithmic ideas are exploited in cryptographic protocols. The new area of blockchains perfectly exemplifies this mixing.

⁵²For this reason it would be impossible to completely elaborate the full assumptions on the models and adversaries; these can be found in the references.

most famously the ALOHA and Ethernet networks in the 1970s [3, 154]. Rabin’s papers, together with the work of Francez and Rodeh [74], are major initial steps in the *formal* design and analysis of probabilistic distributed protocols in the early 1980s.

8.1 Dining philosophers

Edsger Dijkstra, who laid the foundations of distributed computing and concurrent programming, liked to invent stories that abstractly capture the many issues, models, and goals arising in the field. The *dining philosophers* problem is among the most famous of these stories and has served as a model problem for concurrency control ever since. Dijkstra described it in [56], and Hoare formalized it in [110]. This problem abstracts the difficulties of resource allocation in symmetric environments. We repeat Dijkstra’s informal version, which, of course, requires translation into the dry language of processors and resources.

Imagine some philosophers sitting around a table. The philosophers are identical; namely, they have no unique names or IDs. Between each pair of adjacent philosophers is a fork⁵³. In front of every philosopher is a bowl containing an unlimited amount of pasta. Bowls are private, but forks are shared. Philosophers alternate between two activities: thinking and eating. To think, no external resource is needed. To eat, a philosopher must use both forks, the one on her left and the one on her right. The only communication mechanism between (adjacent) philosophers is an *atomic* action that tests whether an adjacent fork is available and, if so, takes it. A philosopher who obtains both forks eats for a while and then releases the forks one after the other.

The original challenge calls for a *deadlock-free* program (in which at least one hungry philosopher eventually gets to eat). A stronger requirement that we will also discuss is a *starvation-free* program (in which every hungry philosopher eventually gets to eat). As the setting is symmetric, we demand that the protocol be symmetric as well. Namely, the program for each philosopher is the same. Initially all philosophers start in the same state and all forks are down.

To illustrate the difficulty, consider the following naive program, which philosophers execute whenever they become hungry.

1. Try to obtain the left fork.
2. Try to obtain the right fork.
3. If unsuccessful in either attempt, return to step 1.
4. Eat.
5. Release the left fork.
6. Release the right fork.

It is clear that this program is not even deadlock-free. Imagine a scheduler that activates each philosopher for just one step (in any order). After this, each is holding her left fork, and this state will never change, regardless of the future activation order. Of course, one can try to avoid deadlock, for example, by modifying the program so that it releases the left fork if the right one is not available. A similar scheduler can easily make them all starve again. Can you find a starvation-free program? There were many suggested programs, but all went outside the model above and either broke symmetry (for example, by giving the philosophers different IDs) or changed the problem

⁵³For this particular story, some people prefer “chopstick” to “fork,” but we stick to the original version.

(for example, by making the philosophers “hygienic” [37], or by adding a “waiter”). But Dijkstra’s original problem remained open.

In 1981, Lehmann and Rabin [140] completely resolved this basic problem, both negatively and positively. They proved three main theorems. The first explains past failures, showing that there is no deterministic solution whatsoever.

Theorem 8.1 (Lehmann–Rabin [140]). *There is no deadlock-free deterministic, symmetric program for the dining philosophers problem.*

A proof must present, for *every* possible symmetric program, a scheduler that causes all philosophers to starve. The idea of their proof is simple (though formalizing it is not) and uses the *same* scheduler for all programs. The scheduler behaves as follows: it activates each philosopher in turn for one step, in cyclic order. This constitutes one phase. It then continues, repeating such phases forever. The invariant maintained inductively on the number of phases is that, before each phase begins, the state of the system is symmetric (i.e., all philosophers are in the same state). This is true at the start, and the symmetry of the program ensures that, in each phase, the result of the (inductively identical) step of every philosopher is the same. To conclude, assume for contradiction that in some phase some philosopher is eating. The inductive claim, and the fact that each philosopher executes exactly the same step with the same result in each phase, imply that, at the end of this phase, all philosophers are eating. But this is impossible, as each needs two forks to eat! Observe that asynchrony is crucial to this impossibility result, but it is used here in a rather benign way. The scheduler essentially prevents any philosopher from executing two successive steps of its program before every other philosopher executes one intervening step.

The next two theorems show how to solve the problem, first for deadlock-freedom and then for starvation-freedom, if we allow programs to be probabilistic.

Theorem 8.2 (Lehmann–Rabin [140, 60]). *There is a probabilistic, symmetric program for the dining philosophers problem that is deadlock-free with probability one.*

Actually, the probabilistic program is a small variation of the naive deterministic program above. Indeed, essentially the only change is that, when a philosopher who holds no forks tries to get one, she tosses a fair coin to decide whether to try the left or the right fork first. The correctness proof is quite nontrivial. The paper makes crucial use of a natural assumption: every philosopher must be activated infinitely often by the scheduler. This assumption was later removed via a modification of the Lehmann–Rabin protocol by DufLOT, Fribourg, and Picaronny [60]. They also developed a very elegant formal framework in which to express the analysis of such probabilistic programs.

We conclude with the final gem from [140]. Lehmann and Rabin observe that, for the probabilistic program above, there do exist schedules that prevent all but one of the philosophers from eating. While deadlock-free, this program is far from satisfactory (at least for hungry philosophers). They then devise a new probabilistic program that fixes this problem and satisfies a stronger *fairness* condition called starvation-freeness. A key ingredient in the new protocol is a slight variation in the information held by the shared variable represented by a fork. The state of a fork now tells us more than whether or not it is in use. It also records, with a couple of extra bits, which of the two adjacent philosophers used it last, as well as the intention of a philosopher to pick it up. Roughly speaking, the resulting probabilistic program (which they call the “courteous philosophers’ algorithm”) prevents the last eater adjacent to a fork from accessing it next, unless the other is not interested. This modification guarantees that, in every schedule, every philosopher who is activated infinitely often will also get to eat infinitely often with probability one.

Theorem 8.3 (Lehmann–Rabin [140]). *There is a probabilistic symmetric program for the dining philosophers problem that is starvation-free with probability one.*

8.2 Mutual exclusion

Another basic resource-allocation problem is mutual exclusion: how to ensure that no more than one of many competing processes can obtain exclusive access to a resource at any given time. The only communication mechanism available to the processes for deciding whether one of them can safely use the resource is a single common *test-and-set* variable: a variable that a single process can *atomically* read and then update in a single step.

For example, assume the variable takes 2 possible values (namely, 1 bit), “free” and “occupied.” A possible protocol for the processes is as follows. Each process that wants to gain access to the resource tries to access the variable. If it succeeds and the value is “free,” it immediately sets it to “occupied,” then uses the resource, and, when done, flips the variable value back to “free.” Regardless of which process is allowed access to this variable at each step, it is clear that mutual exclusion is achieved; no two processes can use the resource simultaneously. Also, it achieves a basic fairness condition, namely, that at least one process needing the resource will get access. However, it may be that two processes need the resource, and scheduling of accesses will allow one to repeatedly use it while the other never does.

What we seek is a stronger notion of fairness, called a *bounded-waiting* protocol for this problem. Such a protocol guarantees that no process trying to gain access to that resource waits forever. As before, the environment is asynchronous, with an arbitrary scheduler dictating the timing of the processes’ actions and access to the shared variable. In this model, processes may be assumed to have unique IDs which can be used when setting the shared variable. Of course, in this case this variable will need to have more states (or bits). The main parameter to minimize is the number of bits that this shared variable should take, as a function of n , the number of processes.

This model was first studied by Burns, Fischer, Jackson, Lynch, and Peterson [33]. They gave a *deterministic* protocol using only $\log n + O(1)$ bits to achieve bounded waiting, and also proved that this is best possible (in other words, the variable must have at least as many states as there are processors). Rabin naturally attempted to beat this bound using randomness. In his paper [195], Rabin devised a probabilistic protocol that reduces this bit-length requirement exponentially, to $O(\log \log n)$, guaranteeing bounded waiting with probability one. Later papers [100, 209] discovered subtle flaws in Rabin’s analysis, which implicitly assumed very weak adversaries. These flaws were corrected, with a modified protocol and new analysis by Kushilevitz and Rabin [135].

Theorem 8.4 (Kushilevitz–Rabin [135]). *Mutual exclusion with bounded waiting can be achieved with probability one by a probabilistic symmetric protocol using only $O(\log \log n)$ bits.*

With Kushilevitz, Mansour, and Zuckerman, Rabin proceeded to prove that the above result is best possible [133].

Theorem 8.5 (Kushilevitz–Mansour–Rabin–Zuckerman [133]). *Any bounded-waiting probabilistic symmetric protocol for mutual-exclusion must use $\Omega(\log \log n)$ bits.*

8.3 Byzantine generals

Another fundamental problem in distributed systems is Byzantine agreement, which captures *coordination* in the presence of faults. Roughly, a collection of processors must agree on a common value, even though some of them may be faulty. The problem was formulated by Pease, Shostak, and Lamport [172, 137]. Their abstraction imagines Byzantine generals in a battle, facing a critical decision of whether to attack or retreat. Each has its own view, but it is paramount that they all agree on a common action. They can send messages (for example, via couriers) to one another to achieve this coordinated decision. However, the problem is complicated by a variety of possible

faults. Messages may be arbitrarily delayed. Also, some generals are faulty: they may die or, worse, be corrupt and send inconsistent messages to different parties. Indeed, the word *Byzantine* is meant to capture the worst possible scenario: an adversary may arbitrarily schedule the arrival of messages and may choose the corrupt generals arbitrarily. Of course, we do not expect anything from the faulty generals, but we want all nonfaulty generals eventually to agree on an action in finite time for any adversary. To prevent a trivial solution in which they all choose an action (for example, “attack”) as a default, regardless of their initial values, we demand that, if they all have the same value, that value must be the chosen action. As usual, we seek a symmetric protocol.

This problem has been studied in both the asynchronous and synchronous models. Two parameters are important here: n , the number of generals, and t , the maximum number of faulty ones. We summarize the results in both, comparing deterministic and probabilistic protocols.

In the asynchronous model, a celebrated result of Fischer, Lynch, and Paterson [72] shows that, even if there is only $t = 1$ fault (and even if this fault is simply a “crash fault,” for example, one general stops communicating at some point), as long as $n \geq 2$, there is no deterministic protocol for this problem. Here again, as with dining philosophers, randomization makes the impossible possible. Ben-Or, soon after obtaining his PhD under Rabin, devised a probabilistic symmetric protocol that can tolerate any $t < n/5$ faults and terminates in finite expected time [19]. This was improved to (the optimal!) $t < n/3$ by Bracha [28]. Unfortunately, both algorithms may take $\exp(n)$ time⁵⁴ to terminate when the number of faults t is linear in n . Breakthrough work of King and Saia [125, 126] devised a protocol that tolerates $t = \Theta(n)$ faults in polynomial expected time. More recently, Huang, Pettie, and Zhu achieved polynomial expected time at the optimal resilience threshold $t < n/3$ [113].

In the synchronous model, there is a global clock for all processors, who proceed in rounds. Messages sent in a round are delivered by the end of that round. Here, deterministic symmetric protocols do exist. However, the number of rounds of communication is essentially the maximum number of faults. The original papers [172, 137] prove that $n > 3t$ is necessary, and give a $(t + 1)$ -round protocol for the problem when this condition holds. The tight lower bound of $t + 1$ on the number of rounds required was proved by Fischer and Lynch [71]. These results are in the information-theoretic model, where the adversary can understand all messages and impersonate any general. Dolev and Strong [58] proved that even in the cryptographic setting, where in particular we allow authenticated messages,⁵⁵ still $t + 1$ rounds are necessary. Clearly, when t is linear in n , this may be impractical.

Rabin shatters this bound using randomization. He devises a symmetric probabilistic protocol that solves the problem in an expected number of rounds that is *constant*, independent of t . A central point (also in Ben-Or’s result above) is that even though the Byzantine adversary can completely observe the protocol, it cannot predict future coin tosses!

Theorem 8.6 (Rabin [196]). *The Byzantine generals problem has a probabilistic protocol that, for every $t < n/10$, terminates in four rounds, in expectation. Moreover, the probability that the number of rounds exceeds any number k is at most $\exp(-k)$.*

The above theorem is actually two theorems, about two protocols. When we allow the generals only crash faults, a simple protocol satisfies the theorem in the information-theoretic setting (Rabin calls this variant the *distributed commit* problem). For arbitrary Byzantine faults, a much more involved protocol satisfies the theorem only in the cryptographic setting (using authentication) and, moreover, makes another strong assumption. Rabin assumes the availability of special correlated

⁵⁴Time must be defined in an asynchronous model; here it is taken to be the length of the longest sequence of message deliveries.

⁵⁵These prevent, for example, A from falsely claiming, “I received message m from B .”

random strings to each processor before the protocol starts⁵⁶. In his paper [196], Rabin poses the challenge of removing this assumption, namely, of obtaining such correlated randomness organically within the protocol.

This challenge was met in the important paper by Feldman and Micali [69]. They use *verifiable secret sharing*, an extension of Shamir’s classical secret-sharing scheme [215], together with a variety of new tools, to give a constant-round protocol to generate shares of such secrets. Thus they remove the assumption in Rabin’s paper, satisfying the theorem in the cryptographic setting⁵⁷ under Byzantine faults; moreover, they improve his tolerance bound to the optimal $t < n/3$.

8.4 Information dispersal

We conclude with another beautiful gem of Rabin. Consider the extremely basic problem, in distributed systems with faults, of transmitting information. Imagine the processors connected through some sparse network N . One processor, A , would like to send a file F of length k to processor B . If there are no faults, the file can be sent along some A – B path in the graph. But in the presence of faults, the failure of any node or edge on that path would cause the information to be lost.

There are many possible fault models, both probabilistic and adversarial. From any such model, we need just one parameter: m , an upper bound on the number of disjoint paths that can fail simultaneously during the transmission of one message.

There is a trivial solution that guarantees safe delivery: send F along $m + 1$ disjoint A – B paths. Surely, one copy of F will arrive intact. Of course, the number of bits sent is now $(m + 1)k$, a factor of m larger, which increases the work and congestion in the network by a factor of m . Can this be avoided?

Rabin’s ingeniously simple solution [198] enables this. Instead of using $m + 1$ paths, he uses $2m$, only a factor of two more. But this enables him to shrink the total amount of information sent to only $2k$, and hence to incur only a factor-two increase in work and congestion, independent of m . Here is how the IDA (Information Dispersal Algorithm) works.

Divide the file F into m disjoint parts, $F_1, F_2, \dots, F_m$, each of size k/m . We can think of each part as an element of a finite field. Now encode this vector of elements, using a linear *erasure code*, as a vector of $2m$ elements $E_1, E_2, \dots, E_{2m}$, such that all the F_i ’s can be recovered from *any* subset of m of the E_j ’s. (We know many such codes, which also have fast encoding and decoding algorithms; Rabin suggested using a Cauchy matrix, but one can also use a Reed–Solomon code, for example.)

Now everything is obvious. A sends each of the E_i ’s along one of the $2m$ disjoint A – B paths. As at most m of the paths will fail, at least m of the E_j ’s will arrive at B , and she will be able to reconstruct the file F from them. Notice that the total amount of information sent is indeed $2k$. Rabin also shows how to implement two variations. One allows the redundancy factor 2 to be reduced to any $1 + \epsilon$, at the cost of using m/ϵ paths. The other allows the use of paths with low congestion (namely every edge is used by only a few of them) instead of completely disjoint paths.

This brings us to an issue we have not yet discussed: do such paths exist, and how does one find them in a distributed manner? This of course depends on the network. Rabin suggests using the Boolean hypercube as the underlying network, and the then-recent *randomized routing* scheme of Valiant [239] to find many low-congestion paths⁵⁸. As these are low-congestion rather than disjoint, Rabin needs to explain how to implement the IDA in this setting. By now, this problem is extremely

⁵⁶Essentially, these are shares, in Shamir’s secret-sharing scheme [215], of random secrets allocated to each processor.

⁵⁷If private links between every pair of players exist, no cryptography is needed.

⁵⁸Another beautiful example of the power of randomness.

well understood; for a variety of highly *expanding* sparse networks there are even deterministic distributed protocols for finding such paths.

Acknowledgements

Many aided me in writing this survey, and I would like to acknowledge and thank them.

On the computer side, I was helped in many ways by ChatGPT 6. I used it in the obvious ways to locate, organize, and summarize material so that I could focus on what I needed to read. I used it to correct typos and grammar, fix L^AT_EX errors, and create the bibliography files. I used it to go over my drafts and suggest revisions on historical, technical, and other grounds. But we also had many fun conversations, and arguments, about the best presentation of the material.

On the human side, many read all or parts of the text and gave me excellent comments that improved it: Michael Ben-Or, Nataly Brukhim, Irit Dinur, David Harel, Eyal Kushilevitz, Nati Linial, Salil Vadhan, Les Valiant, Edna Wigderson, Yuval Wigderson. Thank you all!

References

- [1] Scott Aaronson. Why Philosophers Should Care about Computational Complexity. *Computability: Turing, Gödel, Church, and Beyond*, edited by B. Jack Copeland, Carl J. Posy, and Oron Shagrir, pages 261–328, MIT Press, 2013. (Cited on page 20.)
- [2] Scott Aaronson et al. The Complexity Zoo. Online reference, complexityzoo.net; accessed August 7, 2026. (Cited on page 20.)
- [3] Norman Abramson. The ALOHA System—Another Alternative for Computer Communications. *Proceedings of the November 17–19, 1970, Fall Joint Computer Conference*, volume 37, pages 281–285, 1970. (Cited on page 35.)
- [4] Kehinde Adeogun and Christos A. Kapoutsis. A Quadratic Lower Bound for 2DFAs against One-Way Liveness. [arXiv:2602.24279](https://arxiv.org/abs/2602.24279); an earlier version appeared in *SOFSEM 2026*, pages 104–116, 2026. (Cited on page 13.)
- [5] S. I. Adian. Algorithmic Unsolvability of Problems of Recognition of Certain Properties of Groups. *Doklady Akademii Nauk SSSR*, volume 103, pages 533–535, 1955. (Cited on page 6.)
- [6] Leonard M. Adleman and Ming-Deh A. Huang. Primality Testing and Abelian Varieties over Finite Fields. *Lecture Notes in Mathematics 1512*, Springer, Berlin, 1992. (Cited on page 26.)
- [7] Manindra Agrawal and Somenath Biswas. Primality and Identity Testing via Chinese Remaindering. *Journal of the ACM*, volume 50, number 4, pages 429–443, 2003. (Cited on pages 26 and 33.)
- [8] Manindra Agrawal, Neeraj Kayal, and Nitin Saxena. PRIMES Is in P. *Annals of Mathematics*, volume 160, number 2, pages 781–793, 2004. (Cited on pages 26 and 33.)
- [9] Sanjeev Arora and Boaz Barak. *Computational Complexity: A Modern Approach*. Cambridge University Press, 2009. (Cited on page 3.)
- [10] Association for Computing Machinery. Michael O. Rabin: A.M. Turing Award Laureate. Award citation, shared with Dana S. Scott, 1976. (Cited on page 9.)
- [11] Hagit Attiya and Jennifer Welch. *Distributed Computing: Fundamentals, Simulations, and Advanced Topics*. Wiley, second edition, 2004. (Cited on page 34.)
- [12] Yonatan Aumann, Yan Zong Ding, and Michael O. Rabin. Everlasting Security in the Bounded Storage Model. *IEEE Transactions on Information Theory*, volume 48, number 6, pages 1668–1680, 2002. (Cited on page 33.)
- [13] Yonatan Aumann and Michael O. Rabin. Information Theoretically Secure Communication in the Limited Storage Space Model. *Advances in Cryptology—CRYPTO ’99*, *Lecture Notes in Computer Science 1666*, pages 65–79, Springer, 1999. (Cited on page 33.)
- [14] László Babai. Trading Group Theory for Randomness. *Proceedings of the Seventeenth Annual ACM Symposium on Theory of Computing*, pages 421–429, 1985. (Cited on page 11.)
- [15] László Babai and Endre Szemerédi. On the Complexity of Matrix Group Problems I. *25th Annual Symposium on Foundations of Computer Science*, pages 229–240, IEEE Computer Society, 1984. (Cited on page 8.)

- [16] Bahareh Banyassady and Wolfgang Mulzer. A Simple Analysis of Rabin’s Algorithm for Finding Closest Pairs. *Proceedings of the 33rd European Workshop on Computational Geometry (EuroCG 2017)*, extended abstract, 2017. (Cited on page 27.)
- [17] Ya. M. Barzdin’. Complexity of Recognition of Symmetry on Turing Machines. *Problemy Kibernetiki*, volume 15, pages 245–248; in Russian, 1965. (Cited on page 28.)
- [18] Walter Baur and Michael O. Rabin. Linear Disjointness and Algebraic Complexity. *Logic and Algorithmic: An International Symposium Held in Honour of Ernst Specker*, volume 30 of *Monographies de L’Enseignement Mathématique*, pages 35–46, Geneva, 1982. (Cited on page 21.)
- [19] Michael Ben-Or. Another Advantage of Free Choice: Completely Asynchronous Agreement Protocols. *Proceedings of the Second Annual ACM Symposium on Principles of Distributed Computing*, pages 27–30, 1983. (Cited on page 38.)
- [20] Elwyn R. Berlekamp. Factoring Polynomials over Large Finite Fields. *Mathematics of Computation*, volume 24, number 111, pages 713–735, 1970. (Cited on page 24.)
- [21] Manuel Blum. A Machine-Independent Theory of the Complexity of Recursive Functions. *Journal of the ACM*, volume 14, number 2, pages 322–336, 1967. (Cited on page 19.)
- [22] Manuel Blum. How to Exchange (Secret) Keys. *ACM Transactions on Computer Systems*, volume 1, number 2, pages 175–193; doi:10.1145/357360.357368, 1983. (Cited on page 31.)
- [23] Dan Boneh and Richard J. Lipton. Algorithms for Black-Box Fields and Their Application to Cryptography. *Advances in Cryptology—CRYPTO ’96, Lecture Notes in Computer Science* 1109, pages 283–297, Springer, 1996. (Cited on page 8.)
- [24] William W. Boone. Certain Simple, Unsolvable Problems of Group Theory. V–VI. *Proceedings of the Koninklijke Nederlandse Akademie van Wetenschappen, Series A*, volume 60, pages 22–27 and 227–232, 1957. (Cited on page 6.)
- [25] William W. Boone. The Word Problem. *Annals of Mathematics*, volume 70, number 2, pages 207–265, 1959. (Cited on page 6.)
- [26] Allan Borodin, Stephen A. Cook, and Nicholas Pippenger. Parallel Computation for Well-Endowed Rings and Space-Bounded Probabilistic Machines. *Information and Control*, volume 58, numbers 1–3, pages 113–136, 1983. (Cited on page 23.)
- [27] Robert S. Boyer and J. Strother Moore. A Fast String Searching Algorithm. *Communications of the ACM*, volume 20, number 10, pages 762–772, 1977. (Cited on page 28.)
- [28] Gabriel Bracha. Asynchronous Byzantine Agreement Protocols. *Information and Computation*, volume 75, number 2, pages 130–143, 1987. (Cited on page 38.)
- [29] Dany Breslauer and Zvi Galil. Real-Time Streaming String-Matching. *ACM Transactions on Algorithms*, volume 10, number 4, article 22, 12 pages; <https://doi.org/10.1145/2635814>, 2014. (Cited on page 29.)
- [30] J. Richard Büchi. Weak Second-Order Arithmetic and Finite Automata. *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik*, volume 6, pages 66–92, 1960. (Cited on page 15.)

- [31] J. Richard Büchi. On a Decision Method in Restricted Second Order Arithmetic. *Logic, Methodology and Philosophy of Science: Proceedings of the 1960 International Congress*, pages 1–11, Stanford University Press, 1962. (Cited on pages [16](#) and [17](#).)
- [32] Joe P. Buhler, Jr. Lenstra, Hendrik W., and Carl Pomerance. Factoring Integers with the Number Field Sieve. *The Development of the Number Field Sieve, Lecture Notes in Mathematics 1554*, pages 50–94, Springer; doi:10.1007/BFb0091539, 1993. (Cited on page [30](#).)
- [33] James E. Burns, Paul Jackson, Nancy A. Lynch, Michael J. Fischer, and Gary L. Peterson. Data Requirements for Implementation of N-Process Mutual Exclusion Using a Single Shared Variable. *Journal of the ACM*, volume 29, number 1, pages 183–205, 1982. (Cited on page [37](#).)
- [34] J. Lawrence Carter and Mark N. Wegman. Universal Classes of Hash Functions. *Journal of Computer and System Sciences*, volume 18, number 2, pages 143–154, 1979. (Cited on page [25](#).)
- [35] Jean-Luc Chabert. *A History of Algorithms: From the Pebble to the Microchip*. Springer, Berlin; editor, 1999. (Cited on page [4](#).)
- [36] Ashok K. Chandra, Dexter C. Kozen, and Larry J. Stockmeyer. Alternation. *Journal of the ACM*, volume 28, number 1, pages 114–133, 1981. (Cited on page [11](#).)
- [37] K. Mani Chandy and Jayadev Misra. The Drinking Philosophers Problem. *ACM Transactions on Programming Languages and Systems*, volume 6, number 4, pages 632–646, 1984. (Cited on page [36](#).)
- [38] Noam Chomsky. Three Models for the Description of Language. *IRE Transactions on Information Theory*, volume 2, number 3, pages 113–124, 1956. (Cited on page [10](#).)
- [39] Noam Chomsky. On Certain Formal Properties of Grammars. *Information and Control*, volume 2, number 2, pages 137–167, 1959. (Cited on page [10](#).)
- [40] Noam Chomsky. Context-Free Grammars and Pushdown Storage. *Quarterly Progress Report 65*, Research Laboratory of Electronics, Massachusetts Institute of Technology, pages 187–194, April 1962, 1962. (Cited on pages [10](#) and [15](#).)
- [41] Marek Chrobak. Finite Automata and Unary Languages. *Theoretical Computer Science*, volume 47, number 2, pages 149–158, 1986. (Cited on page [13](#).)
- [42] Marek Chrobak. Errata to: “Finite Automata and Unary Languages”. *Theoretical Computer Science*, volume 302, numbers 1–3, pages 497–498, 2003. (Cited on page [13](#).)
- [43] Alonzo Church. A Note on the Entscheidungsproblem. *The Journal of Symbolic Logic*, volume 1, number 1, pages 40–41, 1936. (Cited on pages [4](#) and [14](#).)
- [44] Alonzo Church. An Unsolvability Problem of Elementary Number Theory. *American Journal of Mathematics*, volume 58, number 2, pages 345–363, 1936. (Cited on page [4](#).)
- [45] Alonzo Church. Review of A. M. Turing, “On Computable Numbers, with an Application to the Entscheidungsproblem”. *The Journal of Symbolic Logic*, volume 2, number 1, pages 42–43, 1937. (Cited on page [4](#).)

- [46] Alonzo Church. Application of Recursive Arithmetic to the Problem of Circuit Synthesis. Summaries of Talks Presented at the Summer Institute for Symbolic Logic, Cornell University, 1957, pages 3–50, Communications Research Division, Institute for Defense Analyses, second edition; presented July 1957 and revised August 1957, 1960. (Cited on page [15](#).)
- [47] Alan Cobham. The Intrinsic Computational Difficulty of Functions. *Logic, Methodology and Philosophy of Science II*, pages 24–30, North-Holland, 1965. (Cited on pages [18](#), [19](#), and [20](#).)
- [48] Anne Condon, Lisa Hellerstein, Samuel Pottle, and Avi Wigderson. On the Power of Finite Automata with Both Nondeterministic and Probabilistic States. *SIAM Journal on Computing*, volume 27, number 3, pages 739–762, 1998. (Cited on page [24](#).)
- [49] Stephen A. Cook. The Complexity of Theorem-Proving Procedures. *Proceedings of the Third Annual ACM Symposium on Theory of Computing*, pages 151–158, 1971. (Cited on page [11](#).)
- [50] Martin Davis. *The Universal Computer: The Road from Leibniz to Turing*. W. W. Norton, New York, 2000. (Cited on page [4](#).)
- [51] Karel de Leeuw, Edward F. Moore, Claude E. Shannon, and Norman Shapiro. Computability by Probabilistic Machines. *Automata Studies*, volume 34 of *Annals of Mathematics Studies*, pages 183–212, Princeton University Press, 1956. (Cited on page [22](#).)
- [52] Max Dehn. Transformation der Kurven auf zweiseitigen Flächen. *Mathematische Annalen*, volume 72, number 3, pages 413–421, 1912. (Cited on page [6](#).)
- [53] Lorenz Demey, Barteld Kooi, and Joshua Sack. Logic and Probability. *The Stanford Encyclopedia of Philosophy*, first published 2013, substantive revision 2023, Metaphysics Research Lab, Stanford University, 2013. (Cited on page [29](#).)
- [54] Richard A. DeMillo and Richard J. Lipton. A Probabilistic Remark on Algebraic Program Testing. *Information Processing Letters*, volume 7, number 4, pages 193–195, 1978. (Cited on page [25](#).)
- [55] Whitfield Diffie and Martin E. Hellman. New Directions in Cryptography. *IEEE Transactions on Information Theory*, volume 22, number 6, pages 644–654, 1976. (Cited on pages [25](#) and [30](#).)
- [56] Edsger W. Dijkstra. Hierarchical Ordering of Sequential Processes. *Acta Informatica*, volume 1, number 2, pages 115–138, 1971. (Cited on page [35](#).)
- [57] Yan Zong Ding and Michael O. Rabin. Hyper-Encryption and Everlasting Security. *Proceedings of STACS 2002*, *Lecture Notes in Computer Science* 2285, pages 1–26, Springer, 2002. (Cited on page [33](#).)
- [58] Danny Dolev and H. Raymond Strong. Authenticated Algorithms for Byzantine Agreement. *SIAM Journal on Computing*, volume 12, number 4, pages 656–666, 1983. (Cited on page [38](#).)
- [59] Doron Drusinsky and David Harel. On the Power of Bounded Concurrency I: Finite Automata. *Journal of the ACM*, volume 41, number 3, pages 517–539; doi:10.1145/176584.176587, 1994. (Cited on page [11](#).)
- [60] Marie Duffot, Laurent Fribourg, and Claudine Picaronny. Randomized Dining Philosophers without Fairness Assumption. *Distributed Computing*, volume 17, number 1, pages 65–76, 2004. (Cited on page [36](#).)

- [61] Cynthia Dwork and Larry Stockmeyer. A Time Complexity Gap for Two-Way Probabilistic Finite-State Automata. *SIAM Journal on Computing*, volume 19, number 6, pages 1011–1023, 1990. (Cited on page [24](#).)
- [62] Stefan Dziembowski and Ueli M. Maurer. Optimal Randomizer Efficiency in the Bounded-Storage Model. *Journal of Cryptology*, volume 17, number 1, pages 5–26; conference version appeared in STOC 2002, 2004. (Cited on page [33](#).)
- [63] Jack Edmonds. Minimum Partition of a Matroid into Independent Subsets. *Journal of Research of the National Bureau of Standards, Section B*, volume 69B, numbers 1–2, pages 67–72; doi:10.6028/jres.069B.004, 1965. (Cited on page [11](#).)
- [64] Jack Edmonds. Paths, Trees, and Flowers. *Canadian Journal of Mathematics*, volume 17, pages 449–467, 1965. (Cited on page [19](#).)
- [65] Jack Edmonds. Optimum Branchings. *Journal of Research of the National Bureau of Standards, Section B*, volume 71B, number 4, pages 233–240; doi:10.6028/jres.071B.032, 1967. (Cited on page [11](#).)
- [66] Samuel Eilenberg. *Automata, Languages, and Machines*. Academic Press, 1974. (Cited on page [14](#).)
- [67] Calvin C. Elgot. Decision Problems of Finite Automata Design and Related Arithmetics. *Transactions of the American Mathematical Society*, volume 98, number 1, pages 21–51, 1961. (Cited on page [15](#).)
- [68] Shimon Even, Oded Goldreich, and Abraham Lempel. A Randomized Protocol for Signing Contracts. *Communications of the ACM*, volume 28, number 6, pages 637–647, 1985. (Cited on pages [31](#) and [33](#).)
- [69] Paul Feldman and Silvio Micali. Optimal Algorithms for Byzantine Agreement. *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, pages 148–161, 1988. (Cited on page [39](#).)
- [70] Michael J. Fischer, Michaela Iorga, and René Peralta. A Public Randomness Service. *Proceedings of the International Conference on Security and Cryptography*, pages 434–438, SciTePress, 2011. (Cited on page [32](#).)
- [71] Michael J. Fischer and Nancy A. Lynch. A Lower Bound for the Time to Assure Interactive Consistency. *Information Processing Letters*, volume 14, number 4, pages 183–186, 1982. (Cited on page [38](#).)
- [72] Michael J. Fischer, Nancy A. Lynch, and Michael S. Paterson. Impossibility of Distributed Consensus with One Faulty Process. *Journal of the ACM*, volume 32, number 2, pages 374–382, 1985. (Cited on page [38](#).)
- [73] Michael J. Fischer and Michael O. Rabin. Super-Exponential Complexity of Presburger Arithmetic. *Complexity of Computation*, volume 7 of SIAM–AMS Proceedings, pages 27–41, American Mathematical Society, 1974. (Cited on page [21](#).)
- [74] Nissim Francez and Michael Rodeh. A Distributed Abstract Data Type Implemented by a Probabilistic Communication Scheme. *21st Annual Symposium on Foundations of Computer Science*, pages 373–379, 1980. (Cited on page [35](#).)

- [75] Rusins Freivalds. Probabilistic Machines Can Use Less Running Time. *Information Processing 77: Proceedings of IFIP Congress 77*, pages 839–842, North-Holland, 1977. (Cited on page [28](#).)
- [76] Rusins Freivalds. Probabilistic Two-Way Machines. *Mathematical Foundations of Computer Science 1981, Lecture Notes in Computer Science 118*, pages 33–45, Springer, 1981. (Cited on page [23](#).)
- [77] Albrecht Fröhlich and John C. Shepherdson. Effective Procedures in Field Theory. *Philosophical Transactions of the Royal Society of London, Series A*, volume 248, number 950, pages 407–432, 1956. (Cited on page [7](#).)
- [78] D. I. Fuchs-Rabinowitsch. Beispiel einer diskreten Gruppe mit endlichvielen Erzeugenden und Relationen, die kein vollständiges System der linearen Darstellungen zulässt. *Doklady Akademii Nauk SSSR*, volume 29, pages 549–550, 1940. (Cited on page [8](#).)
- [79] D. I. Fuchs-Rabinowitsch. Über eine Gruppe mit endlichvielen Erzeugenden und Relationen, die keine isomorphe Darstellung durch Matrizen von endlicher Ordnung zulässt. *Doklady Akademii Nauk SSSR*, volume 27, pages 425–426, 1940. (Cited on page [8](#).)
- [80] Carlo A. Furia. A Survey of Multi-Tape Automata. *arXiv:1205.0178*, 2012. (Cited on page [13](#).)
- [81] David Gale and Frank M. Stewart. Infinite Games with Perfect Information. *Contributions to the Theory of Games II*, volume 28 of *Annals of Mathematics Studies*, pages 245–266, Princeton University Press, 1953. (Cited on page [18](#).)
- [82] Yael Gertner, Sampath Kannan, Tal Malkin, Omer Reingold, and Mahesh Viswanathan. The Relationship between Public Key Encryption and Oblivious Transfer. *Proceedings of the 41st Annual Symposium on Foundations of Computer Science*, pages 325–335; doi:10.1109/SFCS.2000.892121, 2000. (Cited on page [32](#).)
- [83] John Gill. Computational Complexity of Probabilistic Turing Machines. *SIAM Journal on Computing*, volume 6, number 4, pages 675–695, 1977. (Cited on page [25](#).)
- [84] Kurt Gödel. Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme I. *Monatshefte für Mathematik und Physik*, volume 38, pages 173–198, 1931. (Cited on pages [4](#) and [14](#).)
- [85] Oded Goldreich. *Foundations of Cryptography, Volume 1: Basic Tools*. Cambridge University Press, 2001. (Cited on page [29](#).)
- [86] Oded Goldreich. *Foundations of Cryptography, Volume 2: Basic Applications*. Cambridge University Press, 2004. (Cited on page [29](#).)
- [87] Oded Goldreich. *Computational Complexity: A Conceptual Perspective*. Cambridge University Press, 2008. (Cited on page [3](#).)
- [88] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to Play Any Mental Game. *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing*, pages 218–229, 1987. (Cited on page [31](#).)
- [89] Oded Goldreich, Silvio Micali, and Avi Wigderson. Proofs That Yield Nothing but Their Validity or All Languages in NP Have Zero-Knowledge Proof Systems. *Journal of the ACM*, volume 38, number 3, pages 690–728, 1991. (Cited on page [32](#).)

- [90] Shafi Goldwasser and Joe Kilian. Almost All Primes Can Be Quickly Certified. *Proceedings of the Eighteenth Annual ACM Symposium on Theory of Computing*, pages 316–329, 1986. (Cited on page 26.)
- [91] Shafi Goldwasser and Silvio Micali. Probabilistic Encryption and How to Play Mental Poker Keeping Secret All Partial Information. *Proceedings of the Fourteenth Annual ACM Symposium on Theory of Computing*, pages 365–377, 1982. (Cited on pages 25 and 30.)
- [92] Shafi Goldwasser and Silvio Micali. Probabilistic Encryption. *Journal of Computer and System Sciences*, volume 28, number 2, pages 270–299, 1984. (Cited on pages 25 and 30.)
- [93] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The Knowledge Complexity of Interactive Proof Systems. *SIAM Journal on Computing*, volume 18, number 1, pages 186–208, 1989. (Cited on page 11.)
- [94] Erich Grädel, Wolfgang Thomas, and Thomas Wilke. *Automata, Logics, and Infinite Games. Lecture Notes in Computer Science 2500*, Springer; editors, 2002. (Cited on pages 14 and 17.)
- [95] Joseph Y. Halpern. *Reasoning about Uncertainty*. Second edition, MIT Press, 2017. (Cited on page 29.)
- [96] Joseph Y. Halpern and Michael O. Rabin. A Logic to Reason about Likelihood. *Artificial Intelligence*, volume 32, number 3, pages 379–405; doi:10.1016/0004-3702(87)90093-2, 1987. (Cited on page 29.)
- [97] David Harel. *Algorithmics: The Spirit of Computing*. Addison-Wesley, 1987. (Cited on page 3.)
- [98] David Harel and Amir Pnueli. On the Development of Reactive Systems. *Logics and Models of Concurrent Systems*, edited by Krzysztof R. Apt, NATO ASI Series F, volume 13, pages 477–498, Springer; doi:10.1007/978-3-642-82453-1_17, 1985. (Cited on page 15.)
- [99] David Harel, Roni Rosner, and Moshe Y. Vardi. On the Power of Bounded Concurrency III: Reasoning about Programs. *Proceedings of the Fifth Annual IEEE Symposium on Logic in Computer Science*, pages 478–488; doi:10.1109/LICS.1990.113770, 1990. (Cited on page 11.)
- [100] Sergiu Hart, Micha Sharir, and Amir Pnueli. Termination of Probabilistic Concurrent Program. *ACM Transactions on Programming Languages and Systems*, volume 5, number 3, pages 356–380, 1983. (Cited on page 37.)
- [101] Juris Hartmanis. Observations About the Development of Theoretical Computer Science. *Annals of the History of Computing*, volume 3, number 1, pages 42–51, 1981. (Cited on page 18.)
- [102] Juris Hartmanis, II Lewis, Philip M., and Richard E. Stearns. Classification of Computations by Time and Memory Requirements. *Information Processing 1965: Proceedings of IFIP Congress 65*, volume 1, pages 31–35, Spartan Books, 1965. (Cited on page 19.)
- [103] Juris Hartmanis and Richard E. Stearns. On the Computational Complexity of Algorithms. *Transactions of the American Mathematical Society*, volume 117, pages 285–306, 1965. (Cited on pages 18 and 19.)

- [104] Frederick C. Hennie. One-Tape, Off-Line Turing Machine Computations. *Information and Control*, volume 8, number 6, pages 553–578, 1965. (Cited on page 28.)
- [105] David Hilbert. Mathematical Problems. *Bulletin of the American Mathematical Society*, volume 8, number 10, pages 437–479; English translation of the address delivered at the 1900 International Congress of Mathematicians, 1902. (Cited on page 4.)
- [106] David Hilbert and Wilhelm Ackermann. *Grundzüge der theoretischen Logik*. Volume 27 of *Die Grundlehren der mathematischen Wissenschaften*, Julius Springer, Berlin, 1928. (Cited on pages 4 and 14.)
- [107] Tirza Hirst and David Harel. On the Power of Bounded Concurrency II: Pushdown Automata. *Journal of the ACM*, volume 41, number 3, pages 540–554; doi:10.1145/176584.176588, 1994. (Cited on page 11.)
- [108] C. A. R. Hoare. Algorithm 64: Quicksort. *Communications of the ACM*, volume 4, number 7, page 321, 1961. (Cited on page 24.)
- [109] C. A. R. Hoare. Quicksort. *The Computer Journal*, volume 5, number 1, pages 10–16, 1962. (Cited on page 24.)
- [110] C. A. R. Hoare. Communicating Sequential Processes. *Communications of the ACM*, volume 21, number 8, pages 666–677, 1978. (Cited on page 35.)
- [111] John E. Hopcroft and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, 1979. (Cited on page 14.)
- [112] William M. Hoza. Recent Progress on Derandomizing Space-Bounded Computation. *Bulletin of the EATCS*, number 138, pages 114–143; <https://eccc.weizmann.ac.il/report/2022/121/>, 2022. (Cited on page 23.)
- [113] Shang-En Huang, Seth Pettie, and Leqi Zhu. Byzantine Agreement with Optimal Resilience via Statistical Fraud Detection. *Journal of the ACM*, volume 71, number 2, article 12, pages 1–37, 2024. (Cited on page 38.)
- [114] David Israel. Book Review: *The Logic of Mind*. *Computational Linguistics*, volume 11, number 1, 1985. (Cited on page 12.)
- [115] Azad Jiwa, Jennifer Seberry, and Yuliang Zheng. Beacon Based Authentication. *Computer Security—ESORICS 94, Lecture Notes in Computer Science 875*, pages 125–141, Springer, 1994. (Cited on page 32.)
- [116] Erich Kaltofen. Uniform Closure Properties of P-Computable Functions. *Proceedings of the 18th Annual ACM Symposium on Theory of Computing*, pages 330–337; doi:10.1145/12130.12163, 1986. (Cited on page 25.)
- [117] Erich Kaltofen. Factorization of Polynomials Given by Straight-Line Programs. *Randomness and Computation*, edited by Silvio Micali, *Advances in Computing Research* 5, pages 375–412, JAI Press, 1989. (Cited on page 25.)
- [118] Christos A. Kapoutsis. Removing Bidirectionality from Nondeterministic Finite Automata. *Mathematical Foundations of Computer Science 2005, Lecture Notes in Computer Science 3618*, pages 544–555, Springer, 2005. (Cited on page 12.)

- [119] David R. Karger, Philip N. Klein, and Robert E. Tarjan. A Randomized Linear-Time Algorithm to Find Minimum Spanning Trees. *Journal of the ACM*, volume 42, number 2, pages 321–328, 1995. (Cited on page [27](#).)
- [120] Richard M. Karp. Reducibility Among Combinatorial Problems. *Complexity of Computer Computations*, pages 85–103, Plenum Press, 1972. (Cited on page [11](#).)
- [121] Richard M. Karp and Michael O. Rabin. Efficient Randomized Pattern-Matching Algorithms. *IBM Journal of Research and Development*, volume 31, number 2, pages 249–260, 1987. (Cited on page [28](#).)
- [122] Richard M. Karp, Eli Upfal, and Avi Wigderson. Constructing a Perfect Matching is in Random NC. *Combinatorica*, volume 6, number 1, pages 35–48, 1986. (Cited on page [25](#).)
- [123] Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography*. CRC Press, second edition, 2014. (Cited on page [29](#).)
- [124] Joe Kilian. Founding Cryptography on Oblivious Transfer. *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, pages 20–31, 1988. (Cited on page [31](#).)
- [125] Valerie King and Jared Saia. Byzantine Agreement in Polynomial Expected Time. *Proceedings of the Forty-Fifth Annual ACM Symposium on Theory of Computing*, pages 401–410, 2013. (Cited on page [38](#).)
- [126] Valerie King and Jared Saia. Correction to Byzantine Agreement in Expected Polynomial Time, *JACM* 2016. [arXiv:1812.10169](#), 2018. (Cited on page [38](#).)
- [127] Stephen C. Kleene. Representation of Events in Nerve Nets and Finite Automata. Technical Report RM-704, RAND Corporation, Santa Monica, California, December 1951, 1951. (Cited on page [10](#).)
- [128] Stephen C. Kleene. Representation of Events in Nerve Nets and Finite Automata. *Automata Studies*, pages 3–41, Princeton University Press, 1956. (Cited on pages [10](#) and [15](#).)
- [129] Stephen Cole Kleene. *Introduction to Metamathematics*. North-Holland Publishing Company, Amsterdam; also P. Noordhoff and D. Van Nostrand, 1952. (Cited on pages [3](#) and [4](#).)
- [130] Donald E. Knuth, Jr. Morris, James H., and Vaughan R. Pratt. Fast Pattern Matching in Strings. *SIAM Journal on Computing*, volume 6, number 2, pages 323–350, 1977. (Cited on page [28](#).)
- [131] Dexter Kozen. *Automata and Computability*. Springer, 1997. (Cited on page [14](#).)
- [132] Sige-Yuki Kuroda. Classes of Languages and Linear-Bounded Automata. *Information and Control*, volume 7, number 2, pages 207–223, 1964. (Cited on pages [11](#) and [15](#).)
- [133] Eyal Kushilevitz, Yishay Mansour, Michael O. Rabin, and David Zuckerman. Lower Bounds for Randomized Mutual Exclusion. *SIAM Journal on Computing*, volume 27, number 6, pages 1550–1563, 1998. (Cited on page [37](#).)
- [134] Eyal Kushilevitz and Noam Nisan. *Communication Complexity*. Cambridge University Press, 1997. (Cited on page [27](#).)

- [135] Eyal Kushilevitz and Michael O. Rabin. Randomized Mutual Exclusion Algorithms Revisited. *Proceedings of the Eleventh Annual ACM Symposium on Principles of Distributed Computing*, pages 275–283, 1992. (Cited on page [37](#).)
- [136] Joseph-Louis Lagrange. Démonstration d’un théorème d’arithmétique. *Nouveaux Mémoires de l’Académie Royale des Sciences et Belles-Lettres de Berlin*, pages 123–133; volume designated 1770 and published in 1772, 1770. (Cited on page [26](#).)
- [137] Leslie Lamport, Robert Shostak, and Marshall Pease. The Byzantine Generals Problem. *ACM Transactions on Programming Languages and Systems*, volume 4, number 3, pages 382–401, 1982. (Cited on pages [37](#) and [38](#).)
- [138] Clemens Lautemann. BPP and the Polynomial Hierarchy. *Information Processing Letters*, volume 17, number 4, pages 215–217, 1983. (Cited on page [23](#).)
- [139] Hui Huang Lee, Ee-Chien Chang, and Mun Choon Chan. Pervasive Random Beacon in the Internet for Covert Coordination. *Information Hiding, Lecture Notes in Computer Science 3727*, pages 53–61, Springer, 2005. (Cited on page [32](#).)
- [140] Daniel Lehmann and Michael O. Rabin. On the Advantages of Free Choice: A Symmetric and Fully Distributed Solution to the Dining Philosophers Problem. *Proceedings of the 8th ACM SIGPLAN–SIGACT Symposium on Principles of Programming Languages*, pages 133–138, 1981. (Cited on page [36](#).)
- [141] Derrick H. Lehmer. *Computer Technology Applied to the Theory of Numbers*. *Studies in Number Theory*, volume 6 of MAA Studies in Mathematics, pages 117–151, Mathematical Association of America, 1969. (Cited on page [24](#).)
- [142] Gottfried Wilhelm Leibniz. *Dissertatio de Arte Combinatoria*. Fick and Seubold, Leipzig, 1666. (Cited on page [4](#).)
- [143] Leonid A. Levin. Universal Sequential Search Problems. *Problems of Information Transmission*, volume 9, number 3, pages 265–266, 1973. (Cited on page [11](#).)
- [144] Chi-Jen Lu. Encryption against Storage-Bounded Adversaries from On-Line Strong Extractors. *Journal of Cryptology*, volume 17, number 1, pages 27–42, 2004. (Cited on page [34](#).)
- [145] Oleg B. Lupanov. O sravnenii dvukh tipov konechnykh istochnikov. *Problemy Kibernetiki*, volume 9, pages 321–326; English title: A Comparison of Two Types of Finite Sources, 1963. (Cited on page [10](#).)
- [146] Nancy A. Lynch. *Distributed Algorithms*. Morgan Kaufmann, 1996. (Cited on page [34](#).)
- [147] A. A. Markov. On the Impossibility of Certain Algorithms in the Theory of Associative Systems. *Doklady Akademii Nauk SSSR*, volume 55, pages 583–586, 1947. (Cited on page [6](#).)
- [148] Donald A. Martin. Borel Determinacy. *Annals of Mathematics*, volume 102, number 2, pages 363–371, 1975. (Cited on page [18](#).)
- [149] Ueli M. Maurer. Conditionally-Perfect Secrecy and a Provably-Secure Randomized Cipher. *Journal of Cryptology*, volume 5, number 1, pages 53–66, 1992. (Cited on page [33](#).)
- [150] Robert McNaughton. Testing and Generating Infinite Sequences by a Finite Automaton. *Information and Control*, volume 9, number 5, pages 521–530, 1966. (Cited on page [16](#).)

- [151] George H. Mealy. A Method for Synthesizing Sequential Circuits. *Bell System Technical Journal*, volume 34, pages 1045–1079, 1955. (Cited on page [10](#).)
- [152] Luigi Federico Menabrea and Augusta Ada King Lovelace. Sketch of the Analytical Engine Invented by Charles Babbage, Esq., with Notes upon the Memoir by the Translator. *Scientific Memoirs*, volume 3, pages 666–731; translated from Menabrea’s 1842 French article, with extensive notes by Lovelace, 1843. (Cited on page [4](#).)
- [153] George Metakides and Anil Nerode. Effective Content of Field Theory. *Annals of Mathematical Logic*, volume 17, number 3, pages 289–320, 1979. (Cited on page [9](#).)
- [154] Robert M. Metcalfe and David R. Boggs. Ethernet: Distributed Packet Switching for Local Computer Networks. *Communications of the ACM*, volume 19, number 7, pages 395–404, 1976. (Cited on page [35](#).)
- [155] Nicholas Metropolis and Stanislaw Ulam. The Monte Carlo Method. *Journal of the American Statistical Association*, volume 44, number 247, pages 335–341, 1949. (Cited on page [22](#).)
- [156] Albert R. Meyer. Weak Monadic Second-Order Theory of Successor Is Not Elementary-Recursive. *Logic Colloquium, Lecture Notes in Mathematics* 453, pages 132–154, Springer-Verlag, 1975. (Cited on page [16](#).)
- [157] Albert R. Meyer and Larry J. Stockmeyer. The Equivalence Problem for Regular Expressions with Squaring Requires Exponential Space. *13th Annual Symposium on Switching and Automata Theory*, pages 125–129, IEEE Computer Society, 1972. (Cited on pages [11](#) and [21](#).)
- [158] Gary L. Miller. Riemann’s Hypothesis and Tests for Primality. *Journal of Computer and System Sciences*, volume 13, number 3, pages 300–317, 1976. (Cited on page [26](#).)
- [159] Michael Mitzenmacher and Eli Upfal. *Probability and Computing: Randomized Algorithms and Probabilistic Analysis*. Cambridge University Press, 2005. (Cited on page [27](#).)
- [160] Cristopher Moore and Stephan Mertens. *The Nature of Computation*. Oxford University Press, 2011. (Cited on page [3](#).)
- [161] Edward F. Moore. Gedanken-Experiments on Sequential Machines. *Automata Studies*, pages 129–153, Princeton University Press, 1956. (Cited on page [10](#).)
- [162] Rajeev Motwani and Prabhakar Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995. (Cited on page [27](#).)
- [163] Sead Muftić. Transaction Protection by “Antennas”. *Computers and Security*, volume 9, number 3, pages 245–255, 1990. (Cited on page [32](#).)
- [164] Ketan Mulmuley, Umesh V. Vazirani, and Vijay V. Vazirani. Matching Is as Easy as Matrix Inversion. *Combinatorica*, volume 7, number 1, pages 105–113, 1987. (Cited on page [25](#).)
- [165] S. Muthukrishnan. Data Streams: Algorithms and Applications. *Foundations and Trends in Theoretical Computer Science*, volume 1, number 2, pages 117–236, 2005. (Cited on page [29](#).)
- [166] John Myhill. Finite Automata and the Representation of Events. Technical Report WADC 57-624, Wright Air Development Center, 1957. (Cited on page [10](#).)

- [167] Anil Nerode. Linear Automaton Transformations. *Proceedings of the American Mathematical Society*, volume 9, pages 541–544, 1958. (Cited on page 10.)
- [168] P. S. Novikov. On the Algorithmic Unsolvability of the Word Problem in Group Theory. *Trudy Matematicheskogo Instituta imeni V. A. Steklova*, volume 44, pages 1–143, 1955. (Cited on page 6.)
- [169] Derek C. Oppen. A $2^{2^{2^n}}$ Upper Bound on the Complexity of Presburger Arithmetic. *Journal of Computer and System Sciences*, volume 16, number 3, pages 323–332, 1978. (Cited on page 21.)
- [170] Christos H. Papadimitriou. Games Against Nature. *Journal of Computer and System Sciences*, volume 31, number 2, pages 288–301, 1985. (Cited on page 11.)
- [171] Christos H. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994. (Cited on page 3.)
- [172] Marshall C. Pease, Robert E. Shostak, and Leslie Lamport. Reaching Agreement in the Presence of Faults. *Journal of the ACM*, volume 27, number 2, pages 228–234, 1980. (Cited on pages 37 and 38.)
- [173] René C. Peralta. A Simple and Fast Probabilistic Algorithm for Computing Square Roots Modulo a Prime Number. *IEEE Transactions on Information Theory*, volume 32, number 6, pages 846–847, 1986. (Cited on page 24.)
- [174] M. Perles, Michael O. Rabin, and E. Shamir. The Theory of Definite Automata. *IEEE Transactions on Electronic Computers*, volume EC-12, number 3, pages 233–243; essentially identical with Hebrew University Technical Report 6, 1961, 1963. (Cited on pages 19 and 20.)
- [175] Henry C. Pocklington. The Direct Solution of the Quadratic and Cubic Binomial Congruences with Prime Moduli. *Proceedings of the Cambridge Philosophical Society*, volume 19, pages 57–59, 1917. (Cited on page 24.)
- [176] Emil L. Post. Recursive Unsolvability of a Problem of Thue. *Journal of Symbolic Logic*, volume 12, number 1, pages 1–11, 1947. (Cited on page 6.)
- [177] Vaughan R. Pratt. Every Prime Has a Succinct Certificate. *SIAM Journal on Computing*, volume 4, number 3, pages 214–220, 1975. (Cited on page 26.)
- [178] Mojżesz Presburger. Über die Vollständigkeit eines gewissen Systems der Arithmetik ganzer Zahlen, in welchem die Addition als einzige Operation hervortritt. *Comptes Rendus du I Congrès des Mathématiciens des Pays Slaves*, pages 92–101, 1929. (Cited on page 14.)
- [179] Michael O. Rabin. Effective Computability of Winning Strategies. *Contributions to the Theory of Games III*, volume 39 of *Annals of Mathematics Studies*, pages 147–157, Princeton University Press, 1957. (Cited on page 18.)
- [180] Michael O. Rabin. Recursive Unsolvability of Group Theoretic Problems. *Annals of Mathematics*, volume 67, number 1, pages 172–194, 1958. (Cited on pages 5 and 6.)
- [181] Michael O. Rabin. Computable Algebra, General Theory and Theory of Computable Fields. *Transactions of the American Mathematical Society*, volume 95, number 2, pages 341–360, 1960. (Cited on pages 5, 7, 8, and 9.)

- [182] Michael O. Rabin. Computable Algebraic Systems. Summaries of Talks Presented at the Summer Institute for Symbolic Logic, Cornell University, 1957, pages 134–138, Communications Research Division, Institute for Defense Analyses, second edition; talk presented July 1957, 1960. (Cited on page 7.)
- [183] Michael O. Rabin. Degree of Difficulty of Computing a Function and a Partial Ordering of Recursive Sets. Technical Report 2, Hebrew University, Jerusalem, 1960. (Cited on pages 18, 19, and 20.)
- [184] Michael O. Rabin. Probabilistic Automata. *Information and Control*, volume 6, pages 230–245, 1963. (Cited on pages 13, 22, and 23.)
- [185] Michael O. Rabin. Mathematical Theory of Automata. *Mathematical Aspects of Computer Science*, volume 19 of *Proceedings of Symposia in Applied Mathematics*, pages 153–175, American Mathematical Society, 1967. (Cited on page 19.)
- [186] Michael O. Rabin. Decidability of Second-Order Theories and Automata on Infinite Trees. *Transactions of the American Mathematical Society*, volume 141, pages 1–35, 1969. (Cited on pages 14 and 17.)
- [187] Michael O. Rabin. Proving Simultaneous Positivity of Linear Forms. *Journal of Computer and System Sciences*, volume 6, number 6, pages 639–650, 1972. (Cited on page 21.)
- [188] Michael O. Rabin. Solving Linear Equations by Means of Scalar Products. *Complexity of Computer Computations*, pages 11–20, Plenum Press, 1972. (Cited on page 21.)
- [189] Michael O. Rabin. Probabilistic Algorithms. *Algorithms and Complexity: New Directions and Recent Results*, pages 21–39, Academic Press, 1976. (Cited on pages 24, 25, and 26.)
- [190] Michael O. Rabin. Digitalized Signatures and Public-Key Functions as Intractable as Factorization. Technical Report MIT/LCS/TR-212, MIT Laboratory for Computer Science, 1979. (Cited on page 30.)
- [191] Michael O. Rabin. Probabilistic Algorithm for Testing Primality. *Journal of Number Theory*, volume 12, number 1, pages 128–138, 1980. (Cited on page 24.)
- [192] Michael O. Rabin. Probabilistic Algorithms in Finite Fields. *SIAM Journal on Computing*, volume 9, number 2, pages 273–280, 1980. (Cited on page 26.)
- [193] Michael O. Rabin. Fingerprinting by Random Polynomials. Technical Report TR-15-81, Center for Research in Computing Technology, Harvard University, 1981. (Cited on page 28.)
- [194] Michael O. Rabin. How to Exchange Secrets with Oblivious Transfer. Technical Report TR-81, Aiken Computation Laboratory, Harvard University, 1981. (Cited on page 31.)
- [195] Michael O. Rabin. N -Process Mutual Exclusion with Bounded Waiting by $4 \log_2 N$ -Valued Shared Variable. *Journal of Computer and System Sciences*, volume 25, number 1, pages 66–75, 1982. (Cited on page 37.)
- [196] Michael O. Rabin. Randomized Byzantine Generals. 24th Annual Symposium on Foundations of Computer Science, pages 403–409, 1983. (Cited on pages 38 and 39.)
- [197] Michael O. Rabin. Transaction Protection by Beacons. *Journal of Computer and System Sciences*, volume 27, number 2, pages 256–267, 1983. (Cited on pages 32 and 33.)

- [198] Michael O. Rabin. Efficient Dispersal of Information for Security, Load Balancing, and Fault Tolerance. *Journal of the ACM*, volume 36, number 2, pages 335–348, 1989. (Cited on page 39.)
- [199] Michael O. Rabin. A.M. Turing Award Oral History Interview with Michael O. Rabin. Association for Computing Machinery; interview conducted by David Harel in Jerusalem, November 12, 2015, 2015. (Cited on pages 3 and 14.)
- [200] Michael O. Rabin and Dana S. Scott. Finite Automata and Their Decision Problems. *IBM Journal of Research and Development*, volume 3, number 2, pages 114–125, 1959. (Cited on pages 9, 10, 12, and 13.)
- [201] Michael O. Rabin and Jeffrey O. Shallit. Randomized Algorithms in Number Theory. *Communications on Pure and Applied Mathematics*, volume 39, supplement 1, pages S239–S256, 1986. (Cited on page 26.)
- [202] Michael O. Rabin and Shmuel Winograd. Fast Evaluation of Polynomials by Rational Preparation. *Communications on Pure and Applied Mathematics*, volume 25, number 4, pages 433–458, 1972. (Cited on page 21.)
- [203] Prabhakar Raghavan and Clark D. Thompson. Randomized Rounding: A Technique for Provably Good Algorithms and Algorithmic Proofs. *Combinatorica*, volume 7, number 4, pages 365–374, 1987. (Cited on page 25.)
- [204] Henry Gordon Rice. Classes of Recursively Enumerable Sets and Their Decision Problems. *Transactions of the American Mathematical Society*, volume 74, number 2, pages 358–366, 1953. (Cited on page 6.)
- [205] Robert W. Ritchie. Classes of Predictably Computable Functions. *Transactions of the American Mathematical Society*, volume 106, pages 139–173, 1963. (Cited on page 18.)
- [206] Ronald L. Rivest, Adi Shamir, and Leonard Adleman. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *Communications of the ACM*, volume 21, number 2, pages 120–126, 1978. (Cited on pages 25 and 30.)
- [207] John Rompel. One-Way Functions Are Necessary and Sufficient for Secure Signatures. *Proceedings of the Twenty-Second Annual ACM Symposium on Theory of Computing*, pages 387–394, 1990. (Cited on page 32.)
- [208] Shmuel Safra. On the Complexity of ω -Automata. *29th Annual Symposium on Foundations of Computer Science*, pages 319–327, 1988. (Cited on page 16.)
- [209] Isaac Saias. Proving Probabilistic Correctness Statements: The Case of Rabin’s Algorithm for Mutual Exclusion. *Proceedings of the Eleventh Annual ACM Symposium on Principles of Distributed Computing*, pages 263–274, 1992. (Cited on page 37.)
- [210] Jacques Sakarovitch. *Elements of Automata Theory*. Cambridge University Press, 2009. (Cited on page 14.)
- [211] William J. Sakoda and Michael Sipser. Nondeterminism and the Size of Two-Way Finite Automata. *Proceedings of the Tenth Annual ACM Symposium on Theory of Computing*, pages 275–286, 1978. (Cited on page 13.)

- [212] Michael Saks and Shiyu Zhou. $\text{BPHSPACE}(S) \subseteq \text{DSPACE}(S^{3/2})$. *Journal of Computer and System Sciences*, volume 58, number 2, pages 376–403; doi:10.1006/jcss.1998.1616, 1999. (Cited on page 23.)
- [213] Walter J. Savitch. Relationships between Nondeterministic and Deterministic Tape Complexities. *Journal of Computer and System Sciences*, volume 4, number 2, pages 177–192, 1970. (Cited on page 11.)
- [214] Jacob T. Schwartz. Fast Probabilistic Algorithms for Verification of Polynomial Identities. *Journal of the ACM*, volume 27, number 4, pages 701–717, 1980. (Cited on page 25.)
- [215] Adi Shamir. How to Share a Secret. *Communications of the ACM*, volume 22, number 11, pages 612–613, 1979. (Cited on page 39.)
- [216] Adi Shamir. $\text{IP} = \text{PSPACE}$. *Journal of the ACM*, volume 39, number 4, pages 869–877, 1992. (Cited on page 11.)
- [217] Michael Ian Shamos. *Problems in Computational Geometry*. Unpublished book manuscript; revised in 1977 and distributed in photocopy, 1974. (Cited on page 26.)
- [218] Michael Ian Shamos. *Computational Geometry*. Ph.D. thesis, Yale University, New Haven, Connecticut, 1978. (Cited on page 26.)
- [219] Michael Ian Shamos and Dan Hoey. Closest-Point Problems. *16th Annual Symposium on Foundations of Computer Science*, pages 151–162, IEEE, 1975. (Cited on page 26.)
- [220] Claude E. Shannon. Communication Theory of Secrecy Systems. *Bell System Technical Journal*, volume 28, number 4, pages 656–715, 1949. (Cited on page 29.)
- [221] Dennis E. Shasha. An Interview with Michael Rabin. *Communications of the ACM*, volume 53, number 2, pages 37–42, 2010. (Cited on pages 3 and 14.)
- [222] Peter W. Shor. Algorithms for Quantum Computation: Discrete Logarithms and Factoring. *Proceedings of the 35th Annual Symposium on Foundations of Computer Science*, pages 124–134; doi:10.1109/SFCS.1994.365700, 1994. (Cited on page 30.)
- [223] Michael Sipser. A Complexity Theoretic Approach to Randomness. *Proceedings of the Fifteenth Annual ACM Symposium on Theory of Computing*, pages 330–335, 1983. (Cited on page 23.)
- [224] Michael Sipser. *Introduction to the Theory of Computation*. Third edition, Cengage Learning, 2013. (Cited on page 3.)
- [225] Robert Solovay and Volker Strassen. A Fast Monte-Carlo Test for Primality. *SIAM Journal on Computing*, volume 6, number 1, pages 84–85; submitted in 1974, 1977. (Cited on page 24.)
- [226] Larry J. Stockmeyer. The Polynomial-Time Hierarchy. *Theoretical Computer Science*, volume 3, number 1, pages 1–22, 1976. (Cited on page 11.)
- [227] Viggo Stoltenberg-Hansen and John V. Tucker. Computable Rings and Fields. *Handbook of Computability Theory, Studies in Logic and the Foundations of Mathematics* 140, pages 363–447, Elsevier, 1999. (Cited on page 9.)

- [228] Ewa Syta, Philipp Jovanovic, Eleftherios Kokoris Kogias, Nicolas Gailly, Linus Gasser, Ismail Khoffi, Michael J. Fischer, and Bryan Ford. Scalable Bias-Resistant Distributed Randomness. 2017 IEEE Symposium on Security and Privacy, pages 444–460, 2017. (Cited on page 32.)
- [229] Alfred Tarski. A Decision Method for Elementary Algebra and Geometry. University of California Press, 1951. (Cited on page 14.)
- [230] Wolfgang Thomas. On the Synthesis of Strategies in Infinite Games. STACS 1995, Lecture Notes in Computer Science 900, pages 1–13, Springer, 1995. (Cited on pages 14 and 17.)
- [231] Wolfgang Thomas. Languages, Automata, and Logic. Handbook of Formal Languages, volume 3, pages 389–455, Springer, 1997. (Cited on pages 14 and 17.)
- [232] Seinosuke Toda. PP Is as Hard as the Polynomial-Time Hierarchy. SIAM Journal on Computing, volume 20, number 5, pages 865–877, 1991. (Cited on page 23.)
- [233] Boris A. Trakhtenbrot. The Synthesis of Logical Nets Whose Operators Are Described in Terms of Monadic Predicates. Doklady Akademii Nauk SSSR, volume 118, number 4, pages 646–649; in Russian, 1958. (Cited on page 15.)
- [234] Joseph F. Traub. Algorithms and Complexity: New Directions and Recent Results. Academic Press, New York; editor, 1976. (Cited on page 24.)
- [235] A. M. Turing. The Word Problem in Semi-Groups with Cancellation. Annals of Mathematics, volume 52, number 2, pages 491–505, 1950. (Cited on page 6.)
- [236] Alan M. Turing. On Computable Numbers, with an Application to the Entscheidungsproblem. Proceedings of the London Mathematical Society, volume 42, pages 230–265, 1936. (Cited on pages 4, 12, and 14.)
- [237] Alan M. Turing. Computing Machinery and Intelligence. Mind, volume 59, number 236, pages 433–460, 1950. (Cited on page 22.)
- [238] Salil P. Vadhan. Constructing Locally Computable Extractors and Cryptosystems in the Bounded-Storage Model. Journal of Cryptology, volume 17, number 1, pages 43–77; also Cryptology ePrint Archive Report 2002/162, 2004. (Cited on page 34.)
- [239] Leslie G. Valiant. A Scheme for Fast Parallel Communication. SIAM Journal on Computing, volume 11, number 2, pages 350–361, 1982. (Cited on pages 25 and 39.)
- [240] Moshe Y. Vardi and Pierre Wolper. An Automata-Theoretic Approach to Automatic Program Verification. Proceedings of the First Annual Symposium on Logic in Computer Science, pages 332–344, 1986. (Cited on pages 14 and 17.)
- [241] Avi Wigderson. Mathematics and Computation: A Theory Revolutionizing Technology and Science. Princeton University Press, 2019. (Cited on pages 3, 29, and 34.)
- [242] Philip Wolfe. The Strict Determinateness of Certain Infinite Games. Pacific Journal of Mathematics, volume 5, supplement 1, pages 841–847, 1955. (Cited on page 18.)
- [243] Andrew Chi-Chih Yao. Some Complexity Questions Related to Distributive Computing. Proceedings of the Eleventh Annual ACM Symposium on Theory of Computing, pages 209–213, 1979. (Cited on page 27.)

- [244] Andrew Chi-Chih Yao. Protocols for Secure Computations. Proceedings of the 23rd Annual Symposium on Foundations of Computer Science, pages 160–164, 1982. (Cited on page [31](#).)
- [245] Andrew Chi-Chih Yao. How to Generate and Exchange Secrets. Proceedings of the 27th Annual Symposium on Foundations of Computer Science, pages 162–167, 1986. (Cited on page [31](#).)
- [246] Ernst Zermelo. Über eine Anwendung der Mengenlehre auf die Theorie des Schachspiels. Proceedings of the Fifth International Congress of Mathematicians, pages 501–504, 1913. (Cited on page [18](#).)
- [247] Richard Zippel. Probabilistic Algorithms for Sparse Polynomials. Symbolic and Algebraic Computation: EUROSAM 1979, Lecture Notes in Computer Science 72, pages 216–226, Springer, 1979. (Cited on page [25](#).)

A Complete list of Rabin's publications and technical reports

The following list is based on Rabin's Harvard publication list, supplemented and checked where possible against bibliographic databases, publisher records, and institutional archives. Distinct technical-report, conference, and journal versions are retained when they circulated separately or appeared in different years. Entries are ordered by publication year, and alphabetically by title within each year. Each separately listed technical-report, conference, or journal version is placed according to its own year of publication or circulation.

1. Michael O. Rabin. *A theorem on partially ordered sets (Hebrew)*. Riveon Lematematika, vol. 7 (1953), pp. 26-29.
2. Michael O. Rabin. *Sur la représentation des idéaux par des idéaux primaires*. C. R. de l'Académie des sciences, vol. 237 (1953), pp. 544-545.
3. Michael O. Rabin. *On regular polygons with lattice-point vertices (Hebrew)*. Riveon Lematematika, vol. 8 (1954), pp. 13-15.
4. Michael O. Rabin. *A note on Helly's theorem*. Pacific Journal of Mathematics, vol. 5 (1955), pp. 363-366.
5. Michael O. Rabin. *Effective computability of winning strategies*. Annals of Mathematics Studies, vol. 39 (1957), pp. 147-157.
6. Michael O. Rabin. *Two way finite automata*. Proceedings of the Summer Institute of Symbolic Logic, 1957 at Cornell, pp. 366-369.
7. Michael O. Rabin. *Recursive unsolvability of group theoretic problems*. Annals of Mathematics, vol. 67 (1958), pp. 172-194.
8. Michael O. Rabin and Robert Z. Norman. *An algorithm for a minimum cover of a graph*. Proceedings of the American Mathematical Society, vol. 10 (1959), pp. 315-319.
9. Michael O. Rabin. *Arithmetical extensions with prescribed cardinality*. Indagationes Mathematicae, vol. 21 (1959), pp. 439-446.
10. Michael O. Rabin and Dana S. Scott. *Finite automata and their decision problems*. IBM Journal, vol. 3 (1959), pp. 114-125. Russian translation: Konechnye avtomaty i zadachi ikh razresheniya, Kiberneticheskii sbornik, vol. 4 (1962), pp. 58-91. Reprinted in Sequential Machines, Selected Papers, E. F. Moore (editor), Addison-Wesley, Reading, MA (1964), pp. 63-92.
11. Michael O. Rabin and W. Wesley Peterson. *On codes for checking logical operations*. IBM Journal, vol. 3 (1959), pp. 163-168.
12. Michael O. Rabin. *On recursively enumerable and arithmetic models of set theory*. Journal of Symbolic Logic, vol. 23 (1958), appeared 1959, pp. 408-416.
13. Michael O. Rabin. *Speed of computation and classification of recursive sets*. Third Convention of Scientific Societies, Israel (1959), pp. 1-2.
14. Michael O. Rabin. *Computable algebra, general theory and theory of computable fields*. Transactions of the American Mathematical Society, vol. 94 (1960), pp. 341-360.
15. Michael O. Rabin. *Degree of difficulty of computing a function and a partial ordering of recursive sets*. Technical Report O.N.R. Contract (1960), pp. 341-360.
16. Michael O. Rabin. *Non-standard models and the independence of the induction axiom*. Essays on the Foundations of Mathematics, dedicated to A. A. Fraenkel (1961), pp. 287-299.
17. Michael O. Rabin. *Classes of structures and sets of sentences with the intersection property*. Actes du Colloque de mathématiques à l'Occasion du Tricentenaire de la Mort de B. Pascal, Tome 1 (1962), pp. 39-53.
18. Michael O. Rabin. *Diophantine equations and non-standard models of arithmetic*. Proceedings of the International Congress for Logic and Methodology of Science, Stanford (1962), pp. 151-158.

19. Michael O. Rabin. *Probabilistic Automata*. Information and Control, vol. 6 (1963), pp. 230-245. Reprinted in Sequential Machines, Selected Papers, E. F. Moore (editor), Addison-Wesley, Reading, Mass., pp. 98-114.
20. Michael O. Rabin. *Real time computation*. Israel Journal of Mathematics, vol. 1 (1963), pp. 203-211.
21. Micha A. Perles, Michael O. Rabin, and Eli Shamir. *The theory of definite automata*. IEEE Transactions on Electronic Computers, vol. EC-12 (1963), pp. 233-243.
22. Michael O. Rabin and Hao Wang. *Words in the history of a Turing machine with a fixed input*. Journal of the ACM, vol. 10 (1963), pp. 526-527.
23. Michael O. Rabin. *A simple method for undecidability proofs*. Proceedings of the 1964 International Congress for Logic, North-Holland (1965), pp. 58-68.
24. Michael O. Rabin. *Universal groups of automorphisms of models*. Berkeley International Symposium on the Theory of Models, North-Holland (1965), pp. 274-284.
25. Michael O. Rabin. *Classical and probabilistic automata*. Automata Theory, E. B. Cainiello (editor), Academic Press, 1966, pp. 304-313.
26. Calvin C. Elgot and Michael O. Rabin. *Decidability and undecidability of extensions of second (first) order theory of (generalized) successor*. Journal of Symbolic Logic, vol. 31 (1966), pp. 169-181.
27. Michael O. Rabin. *Decidability of Second Order Theories and Automata on Infinite Trees*. Bulletin of the American Mathematical Society, vol. 74 (1968), pp. 1025-1029.
28. Michael O. Rabin. *Mathematical theory of automata*. Proceedings of Symposia in Applied Mathematics, vol. 19, American Mathematical Society, Providence, RI (1968), pp. 153-175.
29. Michael O. Rabin. *Decidability of second-order theories and automata on infinite trees*. Transactions of the American Mathematical Society, vol. 141 (1969), pp. 1-35.
30. Michael O. Rabin. *Automata on infinite trees and the synthesis problem*. ONR Technical Report 37, Jerusalem (1970), 24 pp.
31. Michael O. Rabin. *Weakly definable relations and special automata*. Mathematical Logic and Foundations of Set Theory, Y. Bar-Hillel (editor), North-Holland (1970), pp. 1-23.
32. Michael O. Rabin. *Decidability and definability in second-order theories*. Proceedings of the International Mathematical Congress (1970), published 1971.
33. Michael O. Rabin. *Automata on infinite objects and Church's problem*. Regional Conference Series in Mathematics, No. 13, American Mathematical Society (1972), 22 pp.
34. Michael O. Rabin and Shmuel Winograd. *Fast evaluation of polynomials by rational preparation*. Communications on Pure and Applied Mathematics, vol. 25 (1972), pp. 453-458.
35. Michael O. Rabin. *Proving simultaneous positivity of linear forms*. Journal of Computer and System Sciences, vol. 6 (1972).
36. Michael O. Rabin. *Solving linear equations by means of scalar products*. Complexity of Computer Computations, R. E. Miller (editor), Plenum Press (1972), pp. 11-20.
37. Vaughan R. Pratt, Michael O. Rabin and Larry J. Stockmeyer. *A Characterization of the Power of Vector Machines*. Proceedings of the Sixth Annual ACM Symposium on Theory of Computing (STOC), ACM (1974), pp. 122-134. DOI: [10.1145/800119.803892](https://doi.org/10.1145/800119.803892).
38. Michael J. Fischer and Michael O. Rabin. *Super exponential complexity of Presburger arithmetic*. In Complexity of Computations, SIAM-AMS Proceedings, vol. 7 (1974), pp. 27-41.
39. Michael O. Rabin. *Theoretical impediments to artificial intelligence*. Proceedings of the 1974 IFIP Congress, North-Holland, pp. 615-619.
40. Michael O. Rabin. *Probabilistic algorithms*. In Algorithms and Complexity: New Directions and Recent Trends, J. F. Traub (editor), Academic Press, New York (1976), pp. 21-39.
41. Michael O. Rabin. *Complexity of computations*. Communications of the ACM, vol. 20 (1977), pp. 625-633.

DOI: 10.1145/359810.359816.

42. Michael O. Rabin. *Decidable theories*. Handbook of Mathematical Logic, J. Barwise (editor), North-Holland, Amsterdam (1977), pp. 595-629.
43. Michael O. Rabin. *Digitalized signatures*. Foundations of Secure Computations, R. De Millo and R. Lipton (editors), Academic Press, New York (1978), pp. 155-166.
44. Michael O. Rabin. *Digitalized signatures and public-key functions as intractable as factorization*. MIT/LCS/TR-212, 1979.
45. Walter Baur and Michael O. Rabin. *Linear Disjointness and Algebraic Complexity*. L'Enseignement Mathématique, vol. 26 (1980), pp. 333-344.
46. Michael O. Rabin. *Probabilistic algorithms in finite fields*. SIAM Journal on Computing, vol. 9 (1980), pp. 273-280.
47. Michael O. Rabin. *Probabilistic tests for primality*. Journal of Number Theory, vol. 12 (1980), pp. 128-138.
48. Daniel Lehmann and Michael O. Rabin. *A Symmetric and Fully Distributed Solution to the Dining Philosophers Problem*. (Extended Abstract), 8th ACM Symposium on Principles of Programming Languages (1981), pp. 133-138.
49. Richard M. Karp and Michael O. Rabin. *Efficient Randomized Pattern-Matching Algorithms*. Aiken Computation Laboratory, Harvard University, TR-31-81, Cambridge, MA, 1981. Technical-report version; later journal version appears as RabinPub054. Cited in later systems/security bibliographies as Aiken Laboratory TR-31-81.
50. Michael O. Rabin. *Fingerprinting by random polynomials*. Center for Research in Computing Technology, Harvard University, TR-15-81, 1981.
51. Michael O. Rabin. *How to Exchange Secrets with Oblivious Transfer*. Technical Report TR-81, Aiken Computation Laboratory, Harvard University, 1981; also Cryptology ePrint Archive, Paper 2005/187, 2005.
52. Michael O. Rabin. *N-process mutual exclusion with bounded waiting by $4 \log_2 N$ -valued shared variable*. Journal of Computer and System Sciences, vol. 25 (1982), pp. 66-75.
53. Michael O. Rabin. *The choice coordination problem*. Acta Informatica, vol. 17 (1982), pp. 121-134.
54. Michael O. Rabin. *Randomized Byzantine Generals*. IEEE 24th Symposium on Foundations of Computer Science (1983), pp. 403-409.
55. Michael O. Rabin. *Transaction protection by beacons*. Journal of Computer and System Sciences, vol. 27 (1983).
56. Michael O. Rabin. *Discovering repetitions in strings*. Combinatorial Algorithms on Words, Galil and Apostolico (editors), Springer-Verlag, Berlin CS series (1985), pp. 279-289.
57. Michael O. Rabin and Jeffrey Shallit. *Randomized algorithms in number theory*. Communications on Pure and Applied Mathematics, vol. 39 (1986), pp. 239-256.
58. Joseph Y. Halpern and Michael O. Rabin. *A logic to reason about likelihood*. Artificial Intelligence, vol. 32 (1987), pp. 379-405.
59. Benny Chor and Michael O. Rabin. *Achieving Independence in Logarithmic Number of Rounds*. Proceedings of the 6th ACM Conference on Principles of Distributed Computing, August 1987, pp. 260-268.
60. Michael O. Rabin and J. D. Tygar. *An integrated toolkit for operating system security*. Center for Research in Computing Technology, Harvard University, Technical Report TR-05-87, 1987.
61. Michael O. Rabin. *Efficient Dispersal of Information for Security and Fault Tolerance*. Aiken Computation Laboratory, Harvard University, TR-02-87, Cambridge, MA, 1987. Technical-report version of the information-dispersal work; journal and book-chapter versions appear as RabinPub056 and RabinPub060.
62. Richard M. Karp and Michael O. Rabin. *Efficient randomized pattern-matching algorithms*. IBM Journal of Research and Development, vol. 31 (1987), pp. 249-260.

63. Michael O. Rabin and J. D. Tygar. *An Integrated Toolkit for Operating System Security*. Center for Research in Computing Technology, Aiken Computation Laboratory, Harvard University, TR-05-87R, Cambridge, MA, 1988. Revised technical-report version cited in later bibliographies; related to RabinPub091 and RabinPub059.
64. Michael O. Rabin and J. D. Tygar. *An Integrated Toolkit for Operating System Security*. In Foundations of Data Organization and Algorithms, W. Litwin and H. J. Schek (editors), 3rd International Conference, FODO 1989, Paris, France, June 1989, Springer-Verlag, pp. 2-15.
65. Noga Alon and Michael O. Rabin. *Biased Coins and Randomized Algorithms*. In Randomness and Computing, S. Micali (editor), Advances in Computing, vol. 5, JAI Press (1989), pp. 499-507.
66. Michael O. Rabin. *Efficient Dispersal of Information for Security, Load Balancing, and Fault Tolerance*. Journal of the ACM, vol. 38 (1989), pp. 335-348.
67. Michael O. Rabin and Vijay V. Vazirani. *Maximum Matchings in General Graphs Through Randomization*. Journal of Algorithms, vol. 10 (December 1989).
68. Michael O. Rabin. *The Information Dispersal Algorithm and Its Applications*. In Sequences: Combinatorics, Compression, Security, and Transmission, Capocelli (editor), Springer-Verlag, New York, 1990, pp. 406-419.
69. Noga Alon, Daniel J. Kleitman, Richard J. Lipton, Roy Meshulam, Michael O. Rabin, and Joel Spencer. *Set Systems with No Union of Cardinality 0 Modulo m*. Graphs and Combinatorics, vol. 7 (1991), pp. 97-99.
70. Michael O. Rabin. *Dependable Parallel Computing by Randomization (Abstract)*. Future Tendencies in Computer Science, Control and Applied Mathematics: International Conference on the Occasion of the 25th Anniversary of INRIA, Lecture Notes in Computer Science, vol. 653, Springer (1992), pp. 17-20. DOI: [10.1007/3-540-56320-2_50](https://doi.org/10.1007/3-540-56320-2_50).
71. Zvi M. Kedem, Krishna V. Palem, Michael O. Rabin, and A. Raghunathan. *Efficient Program Transformations for Resilient Parallel Computation via Randomization*. Efficient Program Transformation for Resilient Computation via Randomization, Proceedings of the Annual ACM Symposium on Theory of Computing (STOC), 1992, pp. 306-318.
72. Yonatan Aumann and Michael O. Rabin. *Fast PRAM Simulation on Fully Asynchronous Parallel Systems*. Proceedings of the 33rd Annual Symposium on Foundations of Computer Science (FOCS), October 1992, pp. 147-156.
73. Eyal Kushilevitz and Michael O. Rabin. *Randomized Mutual Exclusion Algorithms Revisited*. Proceedings of Principles of Distributed Computing (PODC), August 1992, pp. 235-283.
74. Yonatan Aumann, Zvi M. Kedem, Krishna V. Palem, and Michael O. Rabin. *Highly efficient asynchronous execution of large-grained parallel programs*. Proceedings of the 34th IEEE Annual Symposium on Foundations of Computer Science (FOCS), 1993, pp. 271-280.
75. Eyal Kushilevitz, Yishay Mansour, Michael O. Rabin, and David Zuckerman. *Lower Bounds for Randomized Mutual Exclusion (Extended Abstract)*. Proceedings of the 25th Annual Symposium on Theory of Computing (STOC), May 16-18, 1993, pp. 154-163.
76. Michael O. Rabin. *Optimal Parallel Pattern Matching Through Randomization*. Sequences II: Methods in Communication, Security, and Computer Science, R. Capocelli et al. (editors), Springer-Verlag, 1993, pp. 292-299.
77. Yonatan Aumann and Michael O. Rabin. *Clock Construction in Fully Asynchronous Parallel Systems and PRAM Simulation*. Theoretical Computer Science, vol. 128 (1994), pp. 3-30.
78. Daniel Lehmann and Michael O. Rabin. *The Advantages of Free Choice: A symmetric and Fully Distributed Solution for the Dining Philosophers Problem*. In A Classical Mind: Essays in Honour of C. A. R. Hoare, A. W. Roscoe (editor), Prentice-Hall International Series in Computer Science (1994), pp. 337-356.
79. Eyal Kushilevitz, Yishay Mansour, and Michael O. Rabin. *On Lotteries with Unique Winners*. SIAM Journal on Discrete Mathematics, vol. 8, no. 1, pp. 93-98, February 1995.

80. Partha Dasgupta, Zvi M. Kedem, and Michael O. Rabin. *Parallel Processing on Networks of Workstations: A Fault-Tolerant, High Performance Approach*. Proceedings of the 15th International Conference on Distributed Computing Systems, 1995 (IEEE Outstanding Paper Award).
81. Michael O. Rabin. *Computationally Hard Algebraic Problems (Invited Lecture)*. Proceedings of the 37th IEEE Annual Symposium on Foundations of Computer Science (FOCS), 1996.
82. Michael O. Rabin. *Correctness of Programs and Protocols through Randomization*. Advances in Computing Science–ASIAN '97, Lecture Notes in Computer Science, vol. 1345, Springer (1997), pp. 1-3. DOI: [10.1007/3-540-63875-X_38](https://doi.org/10.1007/3-540-63875-X_38).
83. Michael O. Rabin and Silvio Micali. *Hashing on Strings, Cryptography, and Protection of Privacy*. Proceedings Compression and Complexity of Sequences, IEEE Computer Society, Los Alamitos, CA, June 11-13, 1997, p. 1.
84. Yonatan Aumann and Michael O. Rabin. *Authentication, Enhanced Security and Error Correcting Codes, IACR Distinguished Lecture*. In Advances in Cryptology–CRYPTO 98, Lecture Notes in Computer Science 1462, Springer, 1998, pp. 299-303.
85. Rosario Gennaro, Michael O. Rabin, and Tal Rabin. *Simplified VSS and Fast-track Multiparty Computations with Applications to Threshold Cryptography*. Proceedings of the 17th ACM Symposium on Principles of Distributed Computing, ACM, 1998, pp. 101-112.
86. Richard J. Lipton, Laura F. Landweber, and Michael O. Rabin. *DNA2DNA Computations: A Potential Killer Application*. In DNA Based Computers III, DIMACS Series in Discrete Mathematics and Theoretical Computer Science, vol. 48, H. Rubin (editor), American Mathematical Society, 1999, pp. 161-172.
87. Yonatan Aumann and Michael O. Rabin. *Information Theoretically Secure Communication in the Limited Storage Model*. In Advances in Cryptology –CRYPTO 99, Lecture Notes in Computer Science 1666, Springer, 1999, pp. 65-79.
88. Silvio Micali, Michael O. Rabin, and Salil P. Vadhan. *Verifiable Random Functions*. Proceedings of the 40th Symposium on Foundations of Computer Science (FOCS), IEEE Computer Society, 1999, pp. 120-130.
89. Yonatan Aumann, Johan Håstad, Michael O. Rabin, and Madhu Sudan. *Linear consistency testing*. Journal of Computer and System Sciences, vol. 62, pp. 589-607, 2001.
90. Yonatan Aumann, Yan Zong Ding, and Michael O. Rabin. *Everlasting Security in the Bounded Storage Model*. IEEE Transactions on Information Theory, vol. 48, issue 6, pp. 1668-1680, June 2002.
91. Yan Zong Ding and Michael O. Rabin. *Hyper-Encryption and Everlasting Security*. Proceedings of the 19th International Symposium on Theoretical Aspects of Computer Science (STACS), pages 1-26, Antibes–Juan les Pins, France, March 14-16, 2002.
92. Michael A. Bender and Michael O. Rabin. *Online Scheduling of Parallel Programs on Heterogeneous Systems with Applications to Cilk*. Theory of Computing Systems, special issue on SPAA '00, vol. 35, pp. 289-304, 2002.
93. Michael O. Rabin. *Hyper Encryption and Everlasting Secrets*. Algorithms and Complexity: Fifth Italian Conference (CIAC), Lecture Notes in Computer Science, vol. 2653, Springer (2003), pp. 7-10. DOI: [10.1007/3-540-44849-7_7](https://doi.org/10.1007/3-540-44849-7_7).
94. Joe Kilian, Silvio Micali, and Michael O. Rabin. *Zero Knowledge Sets*. Symposium on Foundations of Computer Science (FOCS), pp. 80-86, October 2003.
95. Jonathan Katz, Rafail Ostrovsky, and Michael O. Rabin. *Identity-Based Zero Knowledge*. In Security in Communication Networks, 4th International Conference, 2004, Lecture Notes in Computer Science 3352, Springer, 2005, pp. 180-192.
96. Michael O. Rabin and Dennis E. Shasha. *Preventing Piracy While Preserving Privacy*. Dr. Dobb's Journal Computer Security, CMP Media LLC, 2005.
97. Michael O. Rabin. *Provably Unbreakable Hyper-Encryption in the Limited Access Model*. Proceedings

IEEE Symposium on Information Theory Workshop on Theory and Practice in Information-Theoretic Security, pp. 34-37, Awaji Island, Japan, 2005.

98. David C. Parkes, Michael O. Rabin, Stuart M. Shieber, and Christopher Thorpe. *Practical Secrecy-Preserving, Verifiably Correct and Trustworthy Auctions*. ICEC '06: Proceedings of the 8th International Conference on Electronic Commerce, pp. 70-81, New York, NY, USA, ACM Press, 2006.
99. Michael O. Rabin. *DISC 20th Anniversary: Invited Talk Provably Unbreakable Hyper-Encryption Using Distributed Systems*. Distributed Computing: 21st International Symposium (DISC), Lecture Notes in Computer Science, vol. 4731, Springer (2007), pp. 506-508. DOI: [10.1007/978-3-540-75142-7_48](https://doi.org/10.1007/978-3-540-75142-7_48).
100. Michael O. Rabin, Rocco A. Servedio, and Christopher Thorpe. *Highly Efficient Secrecy-Preserving Proofs of Correctness of Computations and Applications*. LICS 2007, Proceedings of the 22nd Annual IEEE Symposium on Logic in Computer Science, 14 pages, 2007.
101. David C. Parkes, Michael O. Rabin, Stuart M. Shieber, and Christopher Thorpe. *Practical Secrecy-Preserving, Verifiably Correct and Trustworthy Auctions*. Electronic Commerce Research and Applications, vol. 7, no. 3, pp. 294-312, 2008.
102. David C. Parkes, Michael O. Rabin and Christopher Thorpe. *Cryptographic Combinatorial Clock-Proxy Auctions*. Financial Cryptography and Data Security: 13th International Conference (FC), Lecture Notes in Computer Science, vol. 5628, Springer (2009), pp. 305-324. DOI: [10.1007/978-3-642-03549-4_19](https://doi.org/10.1007/978-3-642-03549-4_19).
103. Michael O. Rabin. *Never too early to begin: computer science for high-school students*. Proceedings of the 17th ACM Annual Conference on Innovation and Technology in Computer Science Education (ITiCSE), ACM (2012), p. 1. DOI: [10.1145/2325296.2325298](https://doi.org/10.1145/2325296.2325298).
104. Yishay Mansour, S. Muthukrishnan, Michael O. Rabin, and Moti Yung. *Strictly Black-Box Zero-Knowledge and Efficient Validation of Financial Transactions*. ICALP (1) 2012, pp. 738-749.
105. Silvio Micali and Michael O. Rabin. *Cryptography Miracles, Secure Auctions, Matching Problem Verification*. Communications of the ACM, vol. 57, no. 2, pp. 85-93, 2014.