The **BEST** **WRITING** on **MATHEMATICS**

2021

Published by Princeton University Press
41 William Street, Princeton, New Jersey 08540
99 Banbury Road, Oxford OX2 6JX

press.princeton.edu

ISBN 978-0-691-22571-5
ISBN (pbk.) 978-0-691-22570-8
ISBN (ebook) 978-0-691-22572-2

British Library Cataloging-in-Publication Data is available

Editorial: Susannah Shoemaker, Diana Gillooly and Kristen Hop
Production Editorial: Nathan Carr
Text Design: Carmina Alvarez
Cover Design: Chris Ferrante
Production: Jacqueline Poirier
Publicity: Matthew Taylor and Carmen Jimenez
Copyeditor: Paula Bérard

This book has been composed in Perpetua

Printed on acid-free paper. ∞

Printed in the United States of America

1 3 5 7 9 10 8 6 4 2

Contents

Color insert follows page 240

Introduction

Mircea Pitici

The Best Writing on Mathematics 2021 is the twelfth anthology in an annual series bringing together diverse perspectives on mathematics, its applications, and their interpretation—as well as on their social, historical, philosophical, educational, and interdisciplinary contexts. The volume should be seen as a continuation of the previous volumes. Since the series faces an uncertain future, I summarize briefly here its scorecard. We included 293 articles or book chapters in this series, written by almost 400 authors (several authors were represented in the series multiple times), as follows:

BWM Volume	Number of Pieces	Number of Authors
2010	36	43
2011	27	32
2012	25	29
2013	21	23
2014	24	33
2015	29	53
2016	30	41
2017	19	24
2018	18	33
2019	18	32
2020	20	23
2021	26	34
Totals	293	400

The pieces offered this time originally appeared during 2020 in professional publications and/or in online sources. The content of the volume

is the result of a subjective selection process that started with many more candidate articles. I encourage you to explore the pieces that did not make it between the covers of this book; they are listed in the section of notable writings.

This introduction is shorter than the introduction to any of the preceding volumes. I made it a habit to direct the reader to other books on mathematics published recently; this time I will omit that part due to the unprecedented times we lived last year. The libraries accessible to me were closed for much of the research period I dedicated to this volume, and the services for borrowing physical books suffered serious disruptions. A few authors and publishers sent me volumes; yet mentioning here just those titles would be unfair to the many authors whose books I could not obtain.

Overview of the Volume

Once again, this anthology contains an eclectic mix of writings on mathematics, with a few even alluding to the events that just changed our lives in major ways.

To start, Viktor Blåsjö takes a cue from our present circumstances and reviews historical episodes of remarkable mathematical work done in confinement, mostly during wars and in imprisonment.

Andrew Lewis-Pye explains the basic algorithmic rules and computational procedures underlying cryptocurrencies and other blockchain applications, then discusses possible future developments that can make these instruments widely accepted.

Michael Duddy points out that the ascendancy of computational design in architecture leads to an inevitable clash between logic, intellect, and truth on one side—and intuition, feeling, and beauty on the other side. He explains that this trend pushes the decisions traditionally made by the human architect out of the resolutions demanded by the inherent geometry of architecture.

Steve Pomerantz combines elements of basic complex function mapping to reproduce marble mosaic patterns built during the Roman Renaissance of the twelfth and thirteenth centuries.

Ben Logsdon, Anya Michaelsen, and Ralph Morrison construct equations in two variables that represent, in algebraic form, geometric renderings of alphabet letters—thus making it possible to generate

contributions that stand through time, but also insightful in humanistic vision.

Melvyn Nathanson raises the puzzling issues of authorship, copyright, and secrecy in mathematics research, together with many related ethical and practical questions; he comes down uncompromisingly on the side of maximum openness in sharing ideas.

In the end piece of the volume, Terence Tao candidly recalls selected adventures and misadventures of growing into one of the world's foremost mathematicians.

This year has been difficult for all of us; each of us has been affected in one way or another by the current (as of May 2021) health crisis, some tragically. The authors represented in this anthology are no exception. For the first time since the series started, contributors to a volume passed away while the book was in preparation—in this case, John H. Conway (deceased from coronavirus complications) and Harold M. Edwards.

I hope you will enjoy reading this anthology at least as much as I did while working on it. I encourage you to send comments to Mircea Pitici, P.O. Box 4671, Ithaca, NY 14852; or electronic correspondence to mip7@cornell.edu.

Lockdown Mathematics: A Historical Perspective

Viktor Blåsjö

Isolation and Productivity

"A mathematician is comparatively well suited to be in prison." That was the opinion of Sophus Lie, who was incarcerated for a month in 1870. He was 27 at the time. Being locked up did not hamper his research on what was to become Lie groups. "While I was sitting for a month in prison . . ., I had there the best serenity of thought for developing my discoveries," he later recalled [11, pp. 147, 258].

Seventy years later, André Weil was to have a very similar experience. The circumstances of their imprisonments—or perhaps the literary tropes of their retellings—are closely aligned. Having traveled to visit mathematical colleagues, both found themselves engrossed in thought abroad when a war broke out: Lie in France at the outbreak of the Franco-Prussian War, and Weil in Finland at the onset of World War II. They were both swiftly suspected of being spies, due to their strange habits as eccentric mathematicians who incessantly scribbled some sort of incomprehensible notes and wandered in nature without any credible purpose discernible to outsiders. Both were eventually cleared of suspicion upon the intervention of mathematical colleagues who could testify that their behavior was in character for a mathematician and that their mysterious notebooks were not secret ciphers [11, pp. 13–14, 146–147; 13, pp. 130–134].

Weil was deported back to France, where he was imprisoned for another few months for skirting his military duties. Like Lie, he had a productive time in prison. "My mathematics work is proceeding beyond my wildest hopes, and I am even a bit worried—if it's only in prison that I work so well, will I have to arrange to spend two or three months

locked up every year?" "I'm hoping to have some more time here to finish in peace and quiet what I've started. I'm beginning to think that nothing is more conducive to the abstract sciences than prison." "My sister says that when I leave here I should become a monk, since this regime is so conducive to my work."

Weil tells of how colleagues even expressed envy of his prison research retreat. "Almost everyone whom I considered to be my friend wrote me at this time. If certain people failed me then, I was not displeased to discover the true value of their friendship. At the beginning of my time in [prison], the letters were mostly variations on the following theme: 'I know you well enough to have faith that you will endure this ordeal with dignity.' . . . But before long the tone changed. Two months later, Cartan was writing: 'We're not all lucky enough to sit and work undisturbed like you.'" And Cartan was not the only one: "My Hindu friend Vij[ayaraghavan] often used to say that if he spent six months or a year in prison he would most certainly be able to prove the Riemann hypothesis. This may have been true, but he never got the chance."

But Weil grew weary of isolation. He tried to find joy in the little things: "[In the prison yard,] if I crane my neck, I can make out the upper branches of some trees." "When their leaves started to come out in spring, I often recited to myself the lines of the *Gita*: '*Patram puspam phalam toyam* . . .' ('A leaf, a flower, a fruit, water, for a pure heart everything can be an offering')." Soon he was reporting in his letters that "My mathematical fevers have abated; my conscience tells me that, before I can go any further, it is incumbent upon me to work out the details of my proofs, something I find so deadly dull that, even though I spend several hours on it every day, I am hardly getting anywhere" [13, pp. 142–150].

Judging by these examples, then, it would seem that solitary confinement and a suspension of the distractions and obligations of daily life could be very conducive to mathematical productivity for a month or two, but could very well see diminishing returns if prolonged. Of course, it is debatable whether coronavirus lockdown is at all analogous to these gentleman prisons of yesteryear. When Bertrand Russell was imprisoned for a few months for pacifistic political actions in 1918, he too "found prison in many ways quite agreeable. . . . I read enormously; I wrote a book, *Introduction to Mathematical Philosophy*." But his diagnosis

of the cause of this productivity is less relatable, or at least I have yet to hear any colleagues today exclaiming about present circumstances that "the holiday from responsibility is really delightful" [9, pp. 29–30, 32].

Mathematics Shaped by Confinement

"During World War II, Hans Freudenthal, as a Jew, was not allowed to work at the university; it was in those days that his interest in mathematics education at primary school level was sparked by 'playing school' with his children—an interest that was further fueled by conversations with his wife." This observation was made in a recent editorial in *Educational Studies in Mathematics* [1]—a leading journal founded by Hans Freudenthal. Coronavirus lockdown has put many mathematicians in a similar position today. Perhaps we should expect another surge in interest in school mathematics among professional mathematicians.

Freudenthal's contemporary Jakow Trachtenberg, a Jewish engineer, suffered far worse persecution, but likewise adapted his mathematical interests to his circumstances. Imprisoned in a Nazi concentration camp without access to even pen and paper, he developed a system of mental arithmetic. Trachtenberg survived the concentration camp and published his calculation methods in a successful book that has gone through many printings and has its adherents to this day [12].

Another Nazi camp was the birthplace of "spectral sequences and the theory of sheaves . . . by an artillery lieutenant named Jean Leray, during an internment lasting from July 1940 to May 1945." The circumstances of the confinement very much influenced the direction of this research: Leray "succeeded in hiding from the Germans the fact that he was a leading expert in fluid dynamics and mechanics. . . . He turned, instead, to algebraic topology, a field which he deemed unlikely to spawn war-like applications" [10, pp. 41–42].

An earlier case of imprisonment shaping the course of mathematics is Jean-Victor Poncelet's year and a half as a prisoner of war in Russia. Poncelet was part of Napoleon's failed military campaign of 1812 and was only able to return to France in 1814. During his time as a prisoner, he worked on geometry. Poncelet had received a first-rate education in mathematics at the École Polytechnique, and his role in the military was as a lieutenant in the engineering corps. In his Russian prison, he did not have access to any books, so he had to work out

unworkable, "he feigned madness in order to escape the wrath of the Caliph and was confined to a private house for long years until the death of the tyrannical and cruel ruler. He earned his livelihood by copying in secret translations of Euclid's and Ptolemy's works" [7, p. 156]. Euclidean geometry and Ptolemaic astronomical calculations are certainly better suited to house arrest scholarship than engineering projects. One may further wonder whether it is a coincidence that Ibn al-Haytham, who was forced to spend so many sunny days indoors, also discovered the camera obscura and gave it a central role in his optics.

From these examples, we can conclude that if coronavirus measures are set to have an indirect impact on the direction of mathematical research, it would not be the first time lockdown conditions have made one area or style of mathematics more viable than another.

Newton and the Plague

Isaac Newton went into home isolation in 1665, when Cambridge University advised "all Fellows & Scholars" to "go into the Country upon occasion of the Pestilence," since it had "pleased Almighty God in his just severity to visit this towne of Cambridge with the plague" [14, p. 141]. Newton was then 22 and had just obtained his bachelor's degree. His productivity during plague isolation is legendary: this was his *annus mirabilis*, marvelous year, during which he made a number of seminal discoveries. Many have recently pointed to this as a parable for our time, including, for instance, the *Washington Post* [3]. The timeline is none too encouraging for us to contemplate: the university effectively remained closed for nearly two years, with an aborted attempt at reopening halfway through, which only caused "the pestilence" to resurge.

It is true that Newton achieved great things during the plague years, but it is highly doubtful whether the isolation had much to do with it, or whether those years were really all that much more *mirabili* than others. Newton was already making dramatic progress before the plague broke out and was on a trajectory to great discoveries regardless of public health regulations. Indeed, Newton's own account of how much he accomplished "in the two plague years of 1665 & 1666" attributes his breakthroughs not to external circumstances but to his inherent intellectual development: "For in those days I was in the prime of my age for

invention & minded Mathematicks & Philosophy more then at any time since" [15, p. 32].

"Philosophy" here means physics. And indeed, in this subject Newton did much groundwork for his later success during the plague years, but the fundamental vision and synthesis that we associate with Newtonian mechanics today was still distinctly lacking. His eventual breakthrough in physics depended on interactions with colleagues rather than isolation. In 1679, Hooke wrote to Newton for help with the mathematical aspects of his hypothesis "of compounding the celestiall motions of the planetts of a direct motion by the tangent & an attractive motion towards the centrall body." At this time, "Newton was still mired in very confusing older notions." To get Newton going, Hooke had to explicitly suggest the inverse square law and plead that "I doubt not but that by your excellent method you will easily find out what that Curve [the orbit] must be." Only then, "Newton quickly broke through to dynamical enlightenment . . . following [Hooke's] signposted track" [2, pp. 35–37, 117].

Newton later made every effort to minimize the significance of Hooke's role. Indeed, Hooke was just one of many colleagues who ended up on Newton's enemies list. This is another reason why Newton's plague experience is a dubious model to follow. Newton could be a misanthropic recluse even in normal times. When Cambridge was back in full swing, Newton still "seldom left his chamber," contemporaries recalled, except when obligated to lecture—and even that he might as well have done in his chamber for "ofttimes he did in a manner, for want of hearers, read to the walls" [4, n. 11]. He published reluctantly, and when he did, Newton "was unprepared for anything except immediate acceptance of his theory": "a modicum of criticism sufficed, first to incite him to rage, and then to drive him into isolation" [14, pp. 239, 252]. With Hooke, as with so many others, it may well be that Newton only ever begrudgingly interacted with him in the first place for the purpose of proving his own superiority. But that's a social influence all the same. Even if Hooke's role was merely to provoke a sleeping giant, the fact remains that Newton's *Principia* was born then and not in quarantine seclusion.

In mathematics, it is accurate enough to say that Newton "invented calculus" during the plague years. But he was off to a good start already before then, including the discovery of the binomial series. In optics,

Newton himself said that the plague caused a two-year interruption in his experiments on color that he had started while still at Cambridge [6, p. 31]. Perhaps this is another example of pure mathematics being favored in isolation at the expense of other subjects that are more dependent on books and tools.

Home isolation also affords time for extensive hand calculations: a self-reliant mode of mathematics that can be pursued without library and laboratory. Newton did not miss this opportunity during his isolation. As he later recalled, "[before leaving Cambridge] I found the method of Infinite series. And in summer 1665 being forced from Cambridge by the Plague I computed y^e area of y^e Hyperbola . . . to two & fifty figures by the same method" [14, p. 98]. Newton's notebook containing this tedious calculation of the area under a hyperbola to 52 decimals can be viewed at the Cambridge University Library website [8].

References

1 Arthur Bakker and David Wagner, Pandemic: Lessons for today and tomorrow? *Educational Studies in Mathematics*, 104 (2020), 1–4, https://doi.org/10.1007/s10649-020-09946-3.

2 Zev Bechler, Ed., *Contemporary Newtonian Research*, Reidel, Dordrecht, Netherlands, 1982.

3 Gillian Brockell, During a pandemic, Isaac Newton had to work from home, too. He used the time wisely, *Washington Post*, March 12, 2020.

4 I. Bernard Cohen, *Newton, Isaac,* Dictionary of Scientific Biography, Vol. 10, Charles Scribner's Sons, New York, 1974.

5 Isidore Didion, *Notice sur la vie et les ouvrages du Général J.-V. Poncelet*, Gauthier-Villars, Paris, 1869.

6 A. Rupert Hall, *Isaac Newton: Adventurer in Thought*, Cambridge University Press, Cambridge, U.K., 1992.

7 Max Meyerhof, Ali al-Bayhaqi's Tatimmat Siwan al-Hikma: A biographical work on learned men of the Islam, *Osiris* 8 (1948), 122–217.

8 Isaac Newton, MS Add.3958, 79r ff., https://cudl.lib.cam.ac.uk/view/MS-ADD-03958/151.

9 Bertrand Russell, *The Autobiography of Bertrand Russell: 1914–1944*, Little Brown and Company, Boston, 1968.

10 Anna Maria Sigmund, Peter Michor, and Karl Sigmund, Leray in Edelbach. *Mathematical Intelligencer* 27 (2005), 41–50.

11 Arild Stubhaug, *The Mathematician Sophus Lie: It Was the Audacity of My Thinking*, Springer, Berlin, 2002.

12 Jakow Trachtenberg, *The Trachtenberg Speed System of Basic Mathematics*, Doubleday and Company, New York, 1960.

13 André Weil, *The Apprenticeship of a Mathematician*, Birkhäuser, Basel, Switzerland, 1992.

14 Richard S. Westfall, *Never at Rest: A Biography of Isaac Newton*, Cambridge University Press, Cambridge, U.K., 1983.

15 D. T. Whiteside, Newton's Marvellous Year: 1666 and All That, *Notes and Records of the Royal Society of London* 21(1) (1966), 32–41.

Cryptocurrencies: Protocols for Consensus

Andrew Lewis-Pye

The novel feature of Bitcoin [N+08] as a currency is that it is designed to be *decentralized*, i.e., to be run without the use of a central bank, or any centralized point of control. Beyond simply serving as currencies, however, cryptocurrencies like Bitcoin are really protocols for reaching consensus over a decentralized network of users. While running currencies is one possible application of such protocols, one might consider broad swaths of other possible applications. As one example, we have already seen cryptocurrencies used to instantiate *decentralized autonomous organizations* [KOH19], whereby groups of investors come together and coordinate their investments in a decentralized fashion, according to the rules of a protocol that is defined and executed "on the blockchain." One might also envisage new forms of decentralized financial markets, or perhaps even a truly decentralized World Wide Web, in which open-source applications are executed by a community of users, so as to ensure that no single entity (such as Google or Facebook) exerts excessive control over the flow of personal data and other information.

Many questions must be answered before we can talk with any certainty about the extent to which such possibilities can be realized. Some of these questions concern human responses, making the answers especially hard to predict. How much appetite does society have for decentralized applications, and (beyond the possibilities listed above) what might they be? In what contexts will people feel that the supposed advantages of decentralization are worth the corresponding trade-offs in efficiency? There are also basic technical questions to be addressed. Perhaps the best known of these is the so-called *scalability* issue: Can cryptocurrency protocols be made to handle transactions at a rate sufficient to make them useful on a large scale?

In this paper, we will describe how Bitcoin works in simple terms. In particular, this means describing how the Bitcoin protocol uses hard computational puzzles in order to establish consensus as to who owns what. Then we will discuss some of the most significant technical obstacles to the large-scale application of cryptocurrency protocols and approaches that are being developed to solve these problems.

Bitcoin and Nakamoto Consensus

The Bitcoin network launched in January 2009. Since that time, the total value of the currency has been subject to wild fluctuations, but at the time of writing, it is in excess of $170 billion.[1] Given the amount of attention received by Bitcoin, it might be surprising to find out that consensus protocols have been extensively studied in the field of distributed computing since at least the 1970s [Lyn96]. What differentiates Bitcoin from previous protocols, however, is the fact that it is a *permissionless* consensus protocol; i.e., it is designed to establish consensus over a network of users that anybody can join, with as many identities as they like in any role. Anybody with access to basic computational hardware can join the Bitcoin network, and users are often *encouraged* to establish multiple public identities, so that it is harder to trace who is trading with whom.

It is not difficult to see how the requirement for permissionless entry complicates the process of establishing consensus. In the protocols that are traditionally studied in distributed computing, one assumes a fixed set of users, and protocols typically give performance guarantees under the condition that only a certain minority of users behave improperly—"improper" action might include malicious action by users determined to undermine the process. In the permissionless setting, however, users can establish as many identities as they like. Executing a protocol that is only guaranteed to perform well when malicious users are in the minority is thus akin to running an election in which people are allowed to choose their own number of votes.

In the permissionless setting, one therefore needs a mechanism for weighting the contribution of users that goes beyond the system of "one user, one vote." The path taken by Bitcoin is to weight users according to their computational power. This works because computational power is a scarce and testable resource. Users might be able to double

the ledger. Of course, if Alice has to add her signature now, Frank will also have had to add his signature when he transferred the coin to Alice. The signature added to each extension of the ledger can be seen as testimony by the previous owner that they wish to transfer the coin to the new user. The new version of the coin can then be represented as below.

When the central bank sees the new version of the coin, they can check to see that the signature is correct, and, if so, record the transaction as *confirmed*. The use of a signature scheme therefore suffices to ensure that only Alice can spend her coin. This is not the only thing we have to be careful about, though. We also need to be sure that Alice cannot spend her coin twice. In the presence of the central bank, this is also simple. Suppose Alice later creates a new version of the coin, which transfers the coin to another user, Charlie, instead. In this case, the central bank will see that this transaction conflicts with the earlier one that they have seen and so will reject it.

This simple protocol therefore achieves two basic aims:

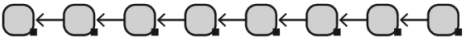
1. Only Alice can spend her coin, and
2. Alice cannot "double spend."

So what changes when we try to do without the use of a central bank? Let us suppose that all users now store a copy of the coin. When Alice wishes to transfer the coin to Bob, she forms a new version of the coin, together with her signature, as before. Now, however, rather than sending it to the central bank, she simply sends the new version to various people in the network of users, who check the signature and then distribute it on to others, and so on. In this case, the use of signatures still suffices to ensure that only Alice can spend her coin. The issue is now that it becomes tricky to ensure that Alice cannot spend her coin twice. Alice could form two new versions of the coin, corresponding to two different transactions. If we could be certain that all other users saw these two versions in the same order, then there would not be a problem, as then users could just agree not to allow the second transaction. Unfortunately, we have no way of ensuring this is the case.[2]

Removing the Central Bank

From the discussion above, it is clear that we need a protocol for establishing irreversible consensus on transaction ordering. To describe how this can be achieved, we will initially describe a protocol that differs from Bitcoin in certain ways, and then we will describe what changes are required to make it the same as Bitcoin later.

Previously, we simplified things by concentrating on one coin. Let us now drop that simplification, and have all users store a *universal ledger*, which records what happens to all coins. We can also drop the simplification that coins are indivisible if we want, and allow transactions which transfer partial units of currency. So according to this modified picture, each user stores a universal ledger, which is just a "chain" of signed transactions. Each transaction in this chain might now follow an unrelated transaction, which transfers a different coin (or part of it) between a different pair of users: The universal ledger is just a chain of transactions recording all transfers of currency that occur between users.



The reader will notice that in the picture above, we have each transaction *pointing to* the previous transaction. We should be clear about how this is achieved, because it is important that we create a tamper-proof ledger: We do not want a malicious user to be able to remove intermediate transactions and produce a version of the universal ledger that looks valid. What we do is to have each signed transaction include the hash of the previous transaction as part of its data. Since hash values are (in effect) unique, this hash value serves as a unique identifier.

What happens next is the key new idea:

(A) We specify a computational puzzle corresponding to each transaction, which is specific to the transaction, and which can be solved only with a lot of computational work. The puzzle is chosen so that, while the solution takes a lot of computational work to find, a correct solution can easily (i.e., efficiently) be verified as correct. The solution to the puzzle corresponding to a given transaction is called a "proof of work" (PoW) for that transaction.

(B) We insist that a transaction cannot be included in the universal ledger unless accompanied by the corresponding PoW.

Do not worry immediately about precisely how the PoW is specified—we will come back to that shortly. Now when Alice wants to spend her coin, she sends the signed transaction out into the network of users, all of whom start trying to produce the necessary PoW. Only once the PoW is found can the transaction be appended to the universal ledger. So now transactions are added to the chain at the rate at which PoWs are found by the network of users. The PoWs are deliberately constructed to require time and resources to complete. Exactly how difficult they are to find is the determining factor in how fast the chain grows.

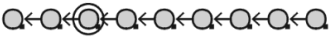
Of course, the danger we are concerned with is that a malicious user might try to form alternative versions of the ledger. How are we to know which version of the ledger is "correct"? In order to deal with these issues, we make two further stipulations (the way in which these stipulations prevent double spending will be explained shortly):

(C) We specify that the "correct" version of the ledger is the longest one. So when users create new transactions, they are asked to have these extend the "longest chain" of transactions (with the corresponding PoWs supplied) they have seen.

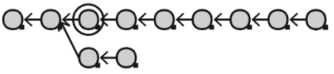
(D) For a certain *security parameter* k, a given user will consider a transaction t as "confirmed" if t belongs to a chain C which is at least k transactions longer than any they have seen that does not include t, and if t is followed by at least k many transactions in C.

The choice of k will depend on how sure one needs to be that double spending does not occur. For the sake of concreteness, the reader might think of $k = 6$ as a reasonable choice.

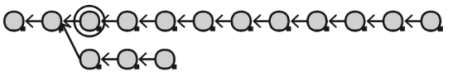
These are quite simple modifications. How do they prevent double spending? The basic idea is as follows. Suppose that at a certain point in time, Alice wants to double spend. Let us suppose that the longest chain of transactions is as depicted below, and that the confirmed transaction t that Alice wants to reverse is the third one (circled).



In order to reverse this transaction, Alice will have to form a new chain that does not include t. This means branching off before t, and building from there.



For people to believe it, however, this new chain will have to be the longest chain. The difficulty for Alice is that while she builds her new chain of transactions, the *rest of the network combined* is working to build the other longer chain.



So long as Alice does not have more computational power than the rest of the network combined, she will not be able to produce PoWs faster than they can. Her chain will therefore grow at a slower rate than the longest chain, and her attempt to double spend will fail.[3] So long as no malicious user (or coordinated set of users) has more power than the rest of the network combined, what we have achieved is a tamper-proof universal ledger, which establishes irreversible consensus on transaction ordering, and which operates in a decentralized way.

To finish this section, we now fulfill some earlier promises. We have to explain how PoWs are defined, what changes are necessary to make the protocol like Bitcoin, and how users come to own coins in the first place.

Defining PoWs

In fact, it will be useful to define PoWs for binary strings more generally—of course, transactions are specified by binary strings of a particular sort. To do this, we fix a good hash function h, and work with a difficulty parameter d, which (is not to be confused with the security parameter k and) can be adjusted to determine how hard the PoW is to find. For two strings x and y, let xy denote the concatenation of x and y. Then we define a PoW for x to be any string y such that $h(xy)$ starts with d many zeros. Given the properties of a good hash function described earlier, this means that there is no more efficient way to find a PoW for x than to plug through possible values for y, requiring 2^d

many attempts on average. The expected time it will take a user to find a PoW is therefore proportional to the rate at which they can process hash values, and for larger d, the PoW will be harder to find. Defining PoW in this way also means that the process by which the network as a whole finds PoWs can reasonably be modeled as a Poisson process: In any second, there is some independent probability that a PoW will be found, and that probability depends on the rate at which the network as a whole can process hashes.

Using Blocks of Transactions

The most significant difference between the protocol we have described and Bitcoin is that in Bitcoin the ledger does not consist of individual transactions, but *blocks* of transactions (hence the term "blockchain"). Each block is a binary string, which contains within its data a few thousand transactions,[4] together with a hash value specifying the previous block. So now, individual transactions are sent out into the network, as before. Rather than requiring a PoW for each individual transaction, however, Bitcoin asks users to collect large sets of transactions into blocks and only requires one PoW per block. The main reason[5] for this is worth understanding properly, because it also relates quite directly to the issue of scalability, which we will discuss in the next section. The key realization here is that we have to take careful account of the fact that the underlying communication network has *latency*; i.e., it takes time for messages to propagate through the network. This latency becomes especially problematic when we work at the level of individual transactions, since they are likely to be produced at a rate that is high compared to network latency. For the sake of concreteness, it may be useful to work with some precise numbers. So, as an example, let us suppose that it takes 10 seconds for a transaction to propagate through the network of users. Suppose that we are using the protocol as defined previously, so that PoWs are required for individual transactions, rather than for blocks. To begin with, let us suppose that the difficulty parameter is set so that the network as a whole finds PoWs for transactions once every 10 minutes on average. Consider a point in time at which all users have seen the same longest chain C, and consider what happens when a PoW for a new transaction t_1 is found by a certain user, so that t_1 can be appended to C. The PoW for t_1 then begins to propagate through

size of blocks, or have them produced more frequently. To increase the transaction rate by a factor of 600, why not have blocks being produced once per second? In fact, the issue here is precisely the same as the motivation for using blocks in the first place, which we discussed in detail previously.[8] Our earlier discussion considered individual transactions, but precisely the same argument holds for blocks of transactions: The fact that the network has *latency* (blocks take a few seconds to propagate through the network) means that whenever a block is produced, there is also the possibility of an honestly produced fork in the blockchain. If we double the rate of block production, then we double the probability of that fork. If we were to have a block produced once per second on average, then we would see forks within forks within forks, and the protocol would no longer be secure.[9] Essentially the same analysis holds in the case that we increase the size of blocks, because doing so increases propagation time. This increase in propagation time similarly increases the probability of a fork.

The processor bottleneck. A basic feature of Bitcoin that distinguishes it from centrally run currencies is that all fully participating users are required to process all transactions. For some applications of blockchain technology, however, one might want to process many millions of transactions per second.[10] To achieve this (even if one solves the latency bottleneck), one needs to deal with the fundamental limitation that transactions can only be processed as fast as can be handled by the slowest user required to process all transactions. The prospect of a decentralized Web 3.0 in which all users have to process all interactions must surely be a nonstarter. So how can one work around this? Limiting the users who have to process all transactions to a small set with such capabilities constitutes a degree of centralization. Another possibility is not to require *any* users to process all transactions. For example, one might consider a process called "sharding," whereby one runs a large number of blockchains that allow limited interactions between them, while requiring each user individually to process transactions on a small set of blockchains at any given time.

Solutions in Three Layers

A multitude of mechanisms have been proposed with the aim of increasing transaction rates. They can be classified as belonging to three *layers*.

Layer 0. These are solutions that do not involve modifying the protocol itself, but aim instead to improve on the underlying infrastructure used by the protocol. Layer 0 solutions range from simply building a faster Internet connection, to approaches such as bloXroute [KBKS18], a blockchain distribution network, which changes the way in which messages propagate through the network. At this point, Layer 0 solutions are generally best seen as approaches to dealing with the latency bottleneck.

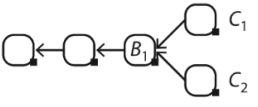
Layer 1. These solutions involve modifying the protocol itself, and they can be aimed at dealing with either the latency bottleneck or the processor bottleneck.

Layer 2. These protocols are implemented *on top of* the underlying cyrptocurrency. So the underlying cryptocurrency is left unchanged, and one runs an extra protocol which makes use of the cryptocurrency's blockchain. Generally, the aim is to outsource work so that most transactions can take place "off-chain," with the underlying cryptocurrency blockchain used (hopefully rarely) to implement conflict resolution. To make these ideas more concrete, we later explain the basic idea behind the Lightning Network, which is probably the best known Layer 2 solution. Layer 2 solutions are generally aimed at solving the processor bottleneck.

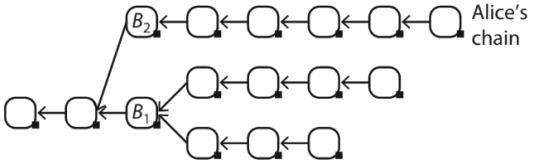
To finish this section, we describe two well-known scalability solutions. Due to the limited available space, we do not say anything further about Layer 0 solutions. We briefly discuss a Layer 1 solution called the GHOST protocol [SZ15], which aims at dealing with the latency bottleneck. Then we explain the basic idea behind the Lightning Network [PD16], already mentioned above as a Layer 2 solution aimed at solving the processor bottleneck.

The GHOST Protocol

Recall that the latency bottleneck was caused by forks: While Bob is waiting for confirmation on a transaction in which Alice sends him money, a fork in the blockchain may split the honest users of the network. Suppose that the transaction is in the block B_1 in the picture below.

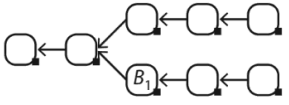


If the honest users are split between chains C_1 and C_2, then these will each grow more slowly than if there was a single chain. This makes it easier for Alice to form a longer chain.



The solution proposed by the GHOST (Greedy Heaviest Observed SubTree) protocol is simple. Rather than selecting the longest chain, we select blocks according to their total number of descendants. This means selecting the chain inductively: Starting with the first block (the so-called "genesis" block), we choose between children by selecting that with the greatest total number of descendants, and then iterate this process to form a longer chain, until we come to a block with no children. This way B_1 will be selected over B_2 in the picture above, because B_1 has seven descendants, while B_2 only has five. So the consequence of using the GHOST protocol is that forks *after* B_1 do not matter, in the sense that they do not change the number of descendants of B_1, and so do not increase Alice's chance of double spending. We can increase the rate of block production, and although there will be an increase in the number of forks, Alice will still require more computational power than the rest of the network combined to double spend.

Unfortunately, however, this modified selection process gives only a partial solution to the latency bottleneck. The reason is that while forks *after* B_1 now do not matter (for confirmation of B_1), forks *before* B_1 still do. To see why, recall that, in order to be confirmed, B_1 must belong to a chain that is longer by some margin than any not including B_1.



If blocks are produced at a rate that is low compared to the time it takes them to propagate through the network, then such (possibly honestly produced) ties are unlikely to persist for long—before too long, an interval of time in which no blocks are produced will suffice to break the tie. If the rate of block production is too fast, however, then

such ties may extend over long periods. This produces long confirmation times.

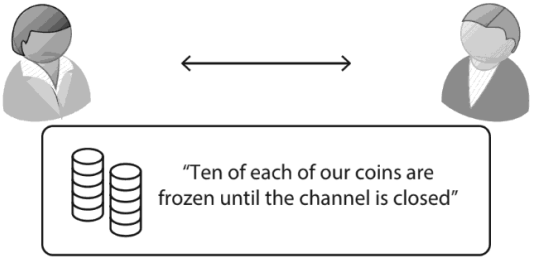
In summary, the GHOST protocol allows us to increase the rate of block production without decreasing the proportion of the network's computational power that Alice will need to double spend. If we increase the rate too much, however, this will result in extended confirmation times.

The Lightning Network

In order to explain the Lightning Network, we first need to discuss "smart contracts."

Smart contracts. So far, we have considered only very simple transactions, in which one user pays another in a straightforward fashion: Alice transfers funds to Bob, in such a way that Bob's signature now suffices to transfer the funds again. Bitcoin does allow, though, for more sophisticated forms of transaction. One might require two signatures to spend money, for example, or perhaps any two from a list of three signatures—so now units of currency might be regarded as having multiple "owners." In such a situation, where there are many forms a transaction could take, how is Alice to specify the transaction she wants to execute? The approach taken by Bitcoin is to use a "scripting language," which allows users to describe how a transaction should work. While Bitcoin has a fairly simple scripting language, other cryptocurrencies, such as Ethereum [W+14], use scripting languages that are sophisticated enough to be *Turing complete*—this means that transactions can be made to simulate any computation in any programming language. As a mathematically minded example, (in principle) one might publish a transaction to the blockchain that automatically pays one million units of currency to anybody who can produce a (suitably encoded) proof of the Riemann hypothesis![11] This is also a functionality whose significance depends on the information available to such computations: If reliable information on stock markets and cryptocurrency prices were to be recorded on the blockchain, then it would immediately become possible to simulate futures, options, and essentially any financial product that can be programmed using the given information. For our purposes now, the point is this: Transactions can be specified to work in much more sophisticated ways than simply transferring currency from one user to another.

A BIDIRECTIONAL PAYMENT CHANNEL. The aim of the Lightning Network is to allow most transactions to take place "off-chain." This is achieved by establishing an auxiliary network of "payment channels." Before coming to the network as a whole, let us consider briefly how to implement an individual channel between two users.[12]



So let us suppose that Alice and Bob wish to set up a payment channel between them. To initiate the channel, they will need to send one transaction to the underlying blockchain. This transaction is signed by both of them and says (in effect) that a certain amount of each of their assets should be frozen until the payment channel is "closed"—closing the channel has a precise meaning that we discuss shortly. For the sake of concreteness, let us suppose that they each freeze 10 coins. Once the channel is set up, Alice and Bob can now trade off-chain, simply by signing a sequence of time-stamped IOUs. If Alice buys something for three coins from Bob, then they both sign a time-stamped IOU stating that Alice owes Bob three coins. If Bob then buys something for one coin from Alice, they both sign a (later) time-stamped IOU stating that Alice now owes Bob two coins. They can continue in this way, so long as neither ever owes the other more than the 10 coins they have frozen. When either user wants to close the channel, they send in the most recent IOU to the blockchain, so that the frozen coins can be distributed to settle the IOU. We must guard against the possibility that the IOU sent is an old one, however. So, once an attempt is made to close the channel, we allow a fixed duration of time for the other user to counter with a more recent IOU.

THE NETWORK. The bidirectional payment channel described above required one transaction in the blockchain to set up, and a maximum of two to close. The system really becomes useful, however, once we have established an extensive network of payment channels.

3. A caveat is that finding a PoW is best modeled as probabilisitic, so there will be some chance that Alice will succeed in double spending, but it will be small.

4. At the time of writing, the monthly mean is just over 2,000 transactions per block.

5. There is a second reason. We want the rate at which PoWs are found, rather than the rate at which users wish to execute transactions, to be the determining factor in how fast the chain grows. One PoW per transaction therefore means requiring a queue of transactions: If there is no queue and if users wish to execute x many transactions each hour, then x many transactions will be added to the chain each hour, and it will be the rate at which users wish to execute transactions that determines how fast the chain grows.

6. It is often asked whether other forms of permissionless blockchain will have more impact than cryptocurrencies. Once one has the latter providing a tamper-proof ledger, this can be used for other applications. Without using a cryptocurrency, however, the task of motivating users to follow protocol will have to be achieved by means other than payment in currency.

7. *See* https://www.cbeci.org/.

8. For a more detailed analysis, we refer the reader to [DW13].

9. Of course, it might still be a good idea to increase the rate by a lower factor.

10. It is a simplification to talk only in terms of the number of transactions. Transaction complexity is also a factor.

11. While this is not presently realistic, it could soon be feasible through the use of smart contracts such as Truebit [TR18].

12. There are a number of ways to implement these details. The Lightning Network is built specifically for Bitcoin, which means that it is designed with the particular functionalities provided by the Bitcoin scripting language in mind. For the sake of simplicity, however, we shall consider building a payment channel on top of a blockchain with a Turing complete scripting language.

References

[BSBHR18] Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev, *Scalable, transparent, and post-quantum secure computational integrity*, 2018. https://eprint.iacr.org/2018/046.

[DW13] Christian Decker and Roger Wattenhofer, *Information propagation in the Bitcoin network*, Ieee p2p 2013 proceedings, 2013, 1–10.

[ES14] Ittay Eyal and Emin Gün Sirer, *Majority is not enough: Bitcoin mining is vulnerable*, International Conference on Financial Cryptography and Data Security, 2014, 436–454.

[GKL15] Juan Garay, Aggelos Kiayias, and Nikos Leonardos, *The Bitcoin backbone protocol: Analysis and applications*, Advances in Cryptology—EUROCRYPT 2015. Part II, Lecture Notes in Comput. Sci., vol. 9057, Springer, Heidelberg, 2015, 281–310, DOI 10.1007/978-3-662-46803-6_10.MR3344957.

[KBKS18] Uri Klarman, Soumya Basu, Aleksandar Kuzmanovic, and Emin Gün Sirer, *bloXroute: A scalable trustless blockchain distribution network whitepaper*, IEEE Internet of Things Journal (2018).

[KOH19] Daniel Kraus, Thierry Obrist, and Olivier Hari, *Blockchains, smart contracts, decentralised autonomous organisations and the law*, Edward Elgar Publishing, 2019.

[LPR20] Andrew Lewis-Pye and Tim Roughgarden, *Resource pools and the cap theorem*, submitted (2020).

[Lyn96] Nancy A. Lynch, *Distributed algorithms*, The Morgan Kaufmann Series in Data Management Systems, Morgan Kaufmann, San Francisco, CA, 1996. MR1388778.

[N+ 08] Satoshi Nakamoto et al., *Bitcoin: A peer-to-peer electronic cash system*, 2008.

[PSS17] Rafael Pass, Lior Seeman, and Abhi Shelat, *Analysis of the blockchain protocol in asynchronous networks*, Advances in Cryptology—EUROCRYPT 2017. Part II, Lecture Notes in Comput. Sci., vol. 10211, Springer, Cham, 2017, 643–673, DOI 10.1007/978-3-319-56614-6. MR3652143.

[PD16] Joseph Poon and Thaddeus Dryja, *The Bitcoin Lightning Network: Scalable off-chain instant payments*, 2016.

[Ren19] Ling Ren, *Analysis of Nakamoto consensus*, Cryptology ePrint Archive, Report 2019/943 (2019). https://eprint.iacr.org.

[SZ15] Yonatan Sompolinsky and Aviv Zohar, *Secure high-rate transaction processing in Bitcoin*, Financial cryptography and data security, Lecture Notes in Comput. Sci., vol. 8975, Springer, Heidelberg, 2015, 507–527, DOI 10.1007/978-3-662-47854-7_32. MR3395041.

[TR18] Jason Teutsch and Christian Reitwießner, *Truebit: A scalable verification solution for blockchains*, 2018.

[W+ 14] Gavin Wood et al., *Ethereum: A secure decentralised generalised transaction ledger*, Ethereum project yellow paper **151** (2014), no. 2014, 1–32.

Logical Accidents and the Problem of the Inside Corner

Michael C. Duddy

Introduction

> Mathematics as an expression of the human mind reflects the active will, the contemplative reason, and the desire for aesthetic perfection. Its basic elements are logic and intuition, analysis and construction, generality and individuality.
>
> —Richard Courant, *What Is Mathematics? An Elementary Approach to Ideas and Methods*

As architectural practice has come to fully embrace digital technology, the algorithmic programs that control the design processes have become an important concern in architectural education. Such algorithms enable the architect to input data as variables into a logical ordering system which in turn outputs representations that correspond to the parameters provided. In fact, the presence of algorithmic methods in the form of rules of design can be traced back to Vitruvius, famously illustrated by the ideal proportions of the human figure.[1] His analogy inspired a full set of formulaic rules for the proper dimensions of the Doric temple that when reinterpreted by Alberti fourteen centuries later provided the rules that impacted architectural practice into the twentieth century. If the rules were followed, according to Alberti, then all parts of the building would correspond and one could determine the proportional relations of the entire building from simply taking the dimensions of an individual part, and the building would embody *perfection*. As computational procedures and artificial intelligence displace human reason in the production of architectural form, will its geometry become merely *exceptional*[2] and epistemologically beyond the capability of human reason—that is, a geometry we cannot understand—leading to

the question of whether future architecture will be capable of achieving a perfection we can apprehend? The complex building geometries made possible by computational design bring relevance to the question of whether there is a geometry inherent to architecture: a geometry that is not applied from outside the discipline of architecture. If so, what distinguishes an architectural geometry from other types of geometry, notably the geometry of mathematics or the geometry of engineering, and what is the foundational logic of this geometry?

Through a close analysis that focuses primarily on the condition of the inside corner, this paper investigates how the logical consequences of a simple system of linear or gridded repetitions that underlie the foundational logic of the architecture that preceded the arrival of the digital turn—a type of *analog algorithm*[3]—lead to complex and seemingly inconsistent conditions when the system meets at the corner. Such consequences are manifested either as *accidents*—visually unresolved conditions that are nevertheless consistent with and conform to the logic of the system—or as *interventions* where the architect violates the system in order to resolve the condition by means of an aesthetic judgment applied from outside the system. Accidents are considered manifestations of the consistency or "truth" of the system, while interventions are understood as inconsistencies imposed on the system through aesthetic judgments. It follows, therefore, that perfection is not a truth but an aesthetic judgment and that such judgments are inherent to the foundational logic of architectural geometry.

In *The Nature and Meaning of Numbers,* the nineteenth-century mathematician, Richard Dedekind, describes a formal system in mathematics as a rule-based discipline that is abstract and rigorous, and whose rules must be executed in a logical order such that no contradictions are encountered and consistency is ensured. Furthermore, the system must stand without meaning, that is, it must be independent of any references to space and time.[4] Similarly, computational designers Achim Menges and Sean Ahlquist define an algorithm in computational design as "a set of procedures consisting of a finite number of rules, which define a succession of operations for the solution of a given problem. In this way, an algorithm is a finite sequence of explicit, elementary instructions described in an exact, complete yet general manner."[5] Accordingly, the foundational logic of computation, just as it is in mathematics, is a discrete set of ordered rules. Yet despite his assertion,

Dedekind acknowledges that there is a point where human intuition participates, as with the reception of an "elegant" proof of modern geometry that satisfies the mind.[6] Because the practice of geometry in architecture is directed toward the harmony of material and space, satisfaction here results when the building appears resolved such that "nothing can be added or taken away except for the worse," as Alberti famously said.

For this study, we will consider the formal system of grids, repetitions, and alternations—a simple system underlying the fundamental modular blocks from which architecture has been traditionally generated. Such a simple system produces what have been described as *metric patterns* that differentiate this simple system from their random environment.[7] For example, the use of identical columns aligned according to a standard interval can be traced back to the earliest buildings where efficient construction required the economy that accompanies repetition. And with the initial publication of Alberti's *De Pictura* in 1433 (1988), the grid entered the architectural canon as the presiding principle to organize experiential space. As Alberti demonstrated, the grid accommodated the orthogonalization of metric space that was essential for the realization of the geometrical construction of linear perspective. Furthermore, Alberti's gridded "veil" provided a framework to translate points in space onto a surface. Henceforth, space could be accurately represented using a rigorous geometrical method and did not rely on the eye of the artist. Tzonis and Lefaivre describe this orthogonalization as an aspect of *taxis*,[8] which, they write, is

> the ordering framework of architecture, divides the building into parts and fits into the resulting partitions the architectural elements, producing a coherent work. . . . . Taxis contains two sublevels, which we call schemata: the grid and the tripartition. The grid schema divides the building through two sets of lines. In the rectangular grid schema, which is one of the most used in classical architecture, straight lines meet at right angles.[9]

They further note that taxis determines the limits of the metric pattern, that the pattern has to begin as well as end at prescribed points.[10] The prescribed points of commencement and termination of a simple repetitive system, the subject of this paper, is the right angle where the system of periodic repetition changes direction, namely, the corner.

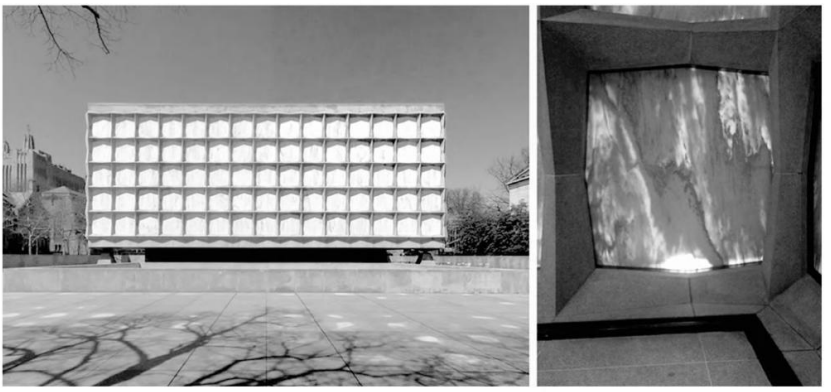


FIGURE 4. To maintain the geometric purity of the Vierendeel grid on the exterior of the Beinecke Library at Yale University, architect Gordon Bunshaft (Skidmore, Owings, and Merrill) had to accommodate an interior corner that seemingly violated the rigor of the grid.

Vierendeel frame. In order for the outside corners to comply exactly with the gridded system, the inside corners are compressed (Figure 4). Clearly in this case, the accident of the inside corner is the result of maintaining the consistency of the outside; the rigor of the system expressed on the outside takes precedence over the visual resolution of the inside corner. Similarly, in Santo Spirito, Brunelleschi takes up the modular system he introduced in the loggia of the Ospedale degli Innocenti. Here both the nave and the transept are organized according to the rule of gridded repetition that maintains a clear logic and consistency throughout the interior as repetitive squares. At the corner on the exterior, however, the semicircular chapels overlap, creating a diagonal condition, which on the exterior yields an odd collision of windows at the inside corner that appears accidental and unresolved (Figure 5). Tzonis and Lefaivre write that "a building made out of a single homogeneous division [such as a grid] runs no risk of violating taxis. Metaphorically we might call it a tautology. It is itself; it contains no element that can contradict it."[13] However, while this may be true of simple planar systems in mathematical geometry, accounting for the material thickness of building inherent to architectural geometry means that either the interior or exterior corner will appear as an accident.