function—the more clearly and directly your work impacts the bottom line, the better.

5. Getting the Second Job

There are a lot of different types of jobs within the finance industry, and your first job is unlikely to be your dream job. However, once in the finance industry, you will quickly meet a lot of other people within the industry. Talk to them and to people in other areas at your firm. Ask them about what they do and what their job is like.

If you discover an area that seems more appealing than the track you are on, moving into it takes some preparation. Figure out what skills are needed for that job and learn them. Even better, find a project in your current role that uses those skills so that you have experience applying them. The timing of the job change is important, too. Moving out of a job too quickly after starting will raise eyebrows, but taking a new job after 2–3 years in your first one is generally seen as acceptable.

Finally, there is no tenure in finance. Turnover is high, and layoffs are common. Every year is a new test, a new chance to prove you can contribute, innovate, and succeed. Good luck!

ACKNOWLEDGMENTS. The author thanks Samer Takriti, Giuseppe Paleologo, Marta Fleming, and an anonymous reviewer for helpful comments on drafts of this article.

Thomas Fleming

Credits

Author photo is courtesy of Thomas Fleming.

A Mathematical Experience in a BIG Career

Kelly B. Yancey

"Intellectual growth should commence at birth and cease only at death."

—Albert Einstein

Research staff members at the Institute for Defense Analyses–Center for Computing Sciences (CCS) are life-long learners. We continue to become proficient in new areas of mathematics and computer science and apply that knowledge to the National Security Agency's mission-research endeavors. It is an exciting and highly collaborative place to work. In addition to high-performance computing for cryptography, CCS's work includes cryptography itself, extensive projects in network security and related cyber issues, signal processing, and emerging algorithmic and mathematical techniques for analyzing extremely complex data sets.

In a 2019 article in the Early Career section of the *Notices* [Yan19] I discussed some of the benefits of working as a research staff member at CCS, a career that I began in August 2016 after I finished a research postdoc at the University of Maryland, College Park (UMD). As I shared in that article, having an open mind about what it means to be a research mathematician has been crucial to my success in finding a satisfying career that works for me and my family.

In 2015, after the second year of my postdoc at UMD, I participated in SCAMP at CCS to learn about a research environment outside of academia. SCAMP is a 10-week summer workshop, hosted every summer by CCS, where researchers tackle difficult problems in support of the National Security Agency's mission. I loved several things about this opportunity: the research environment, my collaborators, and the hard problems that needed to be solved. SCAMP, which brings together mathematicians from a variety of backgrounds and careers, is the subject of a different article in this issue by David Saltman [Sal21].

In my earlier piece and in David Saltman's article, there is advice on how to decide if a career in support of the NSA's mission is right for you. In my 2019 article I also discussed how the chances of being hired by government and industry greatly increase if you prepare ahead of time by participating in summer internships in graduate school or during a postdoc, by taking a few courses in data science or machine learning or statistics, and by learning to program. In this article I highlight the kind of mathematics

Kelly B. Yancey is a research staff member at the Institute for Defense Analyses–Center for Computing Sciences. Her email address is kbyancey1@gmail.com.

DOI: https://dx.doi.org/10.1090/noti2266

that one can do in a BIG career (Business, Industry, and/or Government career) by describing a subset of the results that were unclassified from my SCAMP 2015 experience.

A Collaborative Experience

CCS employs mathematicians trained in a variety of areas: dynamical systems, graph theory, number theory, harmonic analysis, topology, geometry, probability theory, and combinatorics, to name a few. During SCAMP 2015, I worked with mathematicians from a diverse set of backgrounds, and we each brought a unique perspective to the problems at hand. In particular, the work from SCAMP discussed in this article was carried out by Austin Parker (a theoretical computer scientist), Matthew Yancey (an extremal graph theorist), and me (a dynamicist); this research is a blend of all three fields.

During SCAMP 2015, my colleagues and I made precise the problem we wanted to solve, studied literature related to the problem, wrote algorithms and proved related theorems, coded our algorithms, and wrote a paper on our results. The rest of this article is an overview of the parts of that project that were unclassified, as well as research that continued after SCAMP. For more detailed information, see [PYY17, PYY19].

Entropy of a Regular Language

Symbolic dynamics is a field within dynamical systems that has connections with theoretical computer science. In both symbolic dynamics and theoretical computer science, we often analyze infinite languages $\mathcal{L}$ which are composed of finite words over a finite alphabet $\mathcal{A}$. The mathematics discussed in this article illuminates and makes precise the connection between sofic shifts (from symbolic dynamics) and deterministic finite automata (from computer science), and uses that connection to develop a distance between regular languages involving topological entropy.

Regular languages are used in designing programming languages and in many applications such as pattern matching. Do you need to search a document for a keyword? Your computer uses regular languages to accomplish this task via regular expressions. There are many motivations for developing a good distance function to measure similarity between regular languages. Activities in bioinformatics, copy-detection, and network defense sometimes require that large numbers of regular expressions be managed. Metrics aid in indexing and management of those regular expressions.

Given two sets A and B, the symmetric difference $A\Delta B$ is defined as $(\overline{A}\cap B)\cup(A\cap\overline{B})$ and can be used to think about how different the two sets A and B are. In fact, the *Jaccard distance* between two finite sets A and B is the symmetric difference normalized: $\frac{|A\Delta B|}{|A\cup B|}$. In the case of regular languages $\mathcal{L}_1,\mathcal{L}_2$, the symmetric difference $\mathcal{L}_1\Delta\mathcal{L}_2$ is usually infinite.

In [PYY19] we introduce distance functions based on generalizations of Jaccard distance and topological entropy. The distance functions that are related to Jaccard distance are different from prior work by Cui et al. [CDFI13]. In particular, the Cesáro Jaccard distance in [PYY19] is shown to provide a representation of the distance between regular languages that matches intuition and is also proven to be defined for any pair of regular languages. This is in contrast to other Jaccard-type distances for regular languages, including the work by Cui et al., where distances between some pairs of regular languages do not exist.

The distance functions that we define in [PYY19] based on topological entropy are mostly disjoint from Cesáro Jaccard in what they measure. Both types of distance functions can be useful, and for the remainder of this article we focus on a topological entropy-based distance function to highlight the connection between formal language theory and symbolic dynamics.

Background

Regular languages and DFA. Regular languages are exactly those families of words that can be recognized using a finite amount of memory as a proposed word streams by. These languages can be represented by a graphical structure called a deterministic finite automaton (DFA). A *DFA* is a directed graph where the edges are labeled by elements from a finite alphabet $\mathcal{A}$ such that there is at most one outgoing edge from each vertex with a given label, together with an initial state and a set of final states, which are special vertices in the graph. A word is accepted by a DFA if there is a path in the graph associated to that word that begins at the initial state and terminates at a final state. Given this structure, the associated regular language $\mathcal{L}$ is the set of words that are accepted by the DFA [HU79]. Figure 1 provides an example of a DFA over the alphabet $\mathcal{A}=\{a,b\}$ whose associated regular language $\mathcal{L}$ consists of words that contain an even number of a's and an even number of b's. In the parity DFA state 1 is the initial state (indicated by a bold arrow) and the final state (indicated by double circles), and the words *aabb*, *ababbb* $\in\mathcal{L}$.

Performing operations on DFA, including symmetric difference, is a well-studied area; see [HU79] for more information. To measure the complexity of a regular language, we turn to the topological entropy of sofic shifts, a subject in dynamical systems.

Sofic shifts and topological entropy. Sofic shifts, objects in symbolic dynamics, are related to DFA from computer science; see [BP97]. The *full shift* is the set of all bi-infinite sequences over the finite alphabet $\mathcal{A}$ (i.e., $\mathcal{A}^{\mathbb{Z}}=\{(x_i)_{i\in\mathbb{Z}}: x_i\in\mathcal{A}\}$) together with the *shift map* $\sigma:\mathcal{A}^{\mathbb{Z}}\to\mathcal{A}^{\mathbb{Z}}$ defined by $(\sigma(x))_i=x_{i+1}$. A *shift space* (X,σ) is a subset of the full shift that is closed (in the product topology on $\mathcal{A}^{\mathbb{Z}}$) and shift-invariant. A *sofic shift* is a shift space that can be represented by a directed graph where the edges are labeled by elements of $\mathcal{A}$. We assume that all shift spaces are right-solving, which means that there is at most one outgoing edge from each vertex with a given label. A point in a sofic shift space is a bi-infinite walk on the graph.

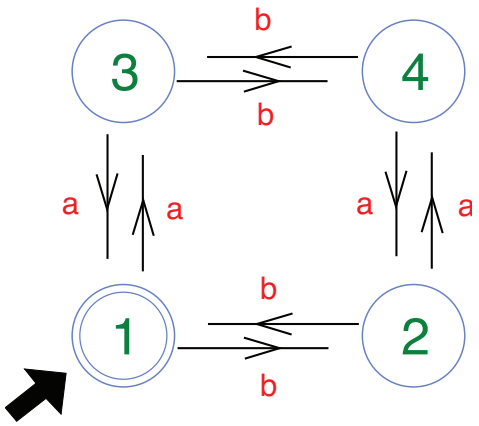

Figure 1. Parity automaton.

The underlying graphical structure of a sofic shift is similar to a DFA; however, there are no initial and final states in a sofic shift. As an example, consider the Golden Mean Shift displayed in Figure 2. In this case, $\mathcal{A} = \{a,b\}$ and the shift space consists of bi-infinite sequences of a's and b's where no two b's are adjacent.

Topological entropy is a concept meant to describe the exponential growth of distinguishable orbits of the dynamical system up to arbitrary scale. A positive quantity for topological entropy reflects chaos in the system. Let (X,σ) be a shift space. We call a finite block *admissible* if there is a point in X where the block appears as a subword. Denote the set of admissible blocks of length n in X by $\mathcal{B}_n(X)$. The *topological entropy* of (X,σ) is given by

$$h_t(X) = \lim_{n\to\infty} \frac{\log|\mathcal{B}_n(X)|}{n}.$$

While topological entropy can be difficult to compute in general, it is an easy calculation for sofic shifts. Given a directed graph G on m vertices, the corresponding *adjacency matrix* A is an $m \times m$ matrix where the ijth entry $A_{(ij)}$ is the number of edges from vertex i to vertex j. Using generalizations of Perron-Frobenius theory, it has been proven that the topological entropy of a sofic shift represented by a labeled graph G is equal to the log of the spectral radius of the adjacency matrix of G. That is, the topological entropy is given by the log of the adjacency matrix's largest modulus eigenvalue. Algorithms for computing eigenvalues are well known and run in polynomial time in the width of the matrix. For more information on the topological entropy of sofic shifts, see [LM95].

As an example, we will compute the topological entropy of the Golden Mean Shift, X, in Figure 2. The adjacency matrix for this shift is given by $\left(\begin{smallmatrix} 1 & 1 \\ 1 & 0 \end{smallmatrix}\right)$. The eigenvalues of this matrix are $\frac{1\pm\sqrt{5}}{2}$, which means that $h_t(X) = \log(\frac{1+\sqrt{5}}{2})$ (hence the name Golden Mean Shift).

An *irreducible component* of a directed graph G is a maximal subgraph H of G such that for every pair of vertices u and v in H, there is a directed path from u to v. Irreducible components can be computed in linear time. The topological entropy of a sofic shift X represented by a graph G can be computed by taking the maximum of the topological entropies of sofic shifts corresponding to irreducible components of G.

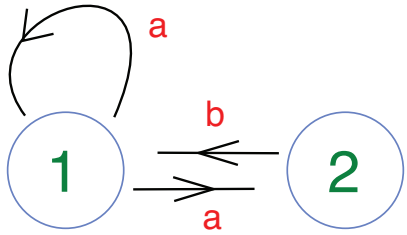

Figure 2. The Golden Mean Shift.

Topological Entropy vs. Language Entropy

In their foundational paper on regular languages [CM58], Chomsky and Miller define the entropy of a regular language $\mathcal{L}$ as Shannon's *channel capacity*:

$$\lim_{n\to\infty} \frac{\log|W_n(\mathcal{L})|}{n}$$

where $W_n(\mathcal{L})$ is the set of words in $\mathcal{L}$ of length exactly n. Chomsky and Miller's technique was to develop a recursive formula for the number of words accepted by a regular language. That recursive formula comes from the characteristic polynomial of the adjacency matrix for an associated automaton. The eigenvalues of the adjacency matrix describe the growth of the language. However, Chomsky and Miller incorrectly applied the Perron-Frobenius theorem in their proof of convergence; this limit does not always exist for regular languages, as seen in the language consisting of even length words.

The formal language community has adopted the convention of defining the entropy of a regular language as

$$\limsup_{n\to\infty} = \frac{\log|W_n(\mathcal{L})|}{n}$$

to address the issue of convergence. Much of the analysis in [PYY19] is devoted to constructing a definition of entropy of a regular language that describes the overall growth of the language while still being rigorous and matching existing intuition. We prove that the upper limit in the channel capacity is realized by the topological entropy of the corresponding sofic shift and define another notion of entropy, which is preferable since an upper limit is not necessary. We use similar techniques to those employed by Chomsky and Miller; however, we apply generalizations of Perron-Frobenius theory that were discovered by Rothblum several decades after the work of Chomsky and Miller on entropy [Rot81]. For a regular language $\mathcal{L}$ define the *language entropy* to be

$$h(\mathcal{L}) = \lim_{n\to\infty} \frac{\log|W_{\leq n}(\mathcal{L})|}{n}$$

where $\mathcal{W}_{\leq n}(\mathcal{L})$ is the set of words in L of length *at most* n. Given a DFA, the associated sofic shift will be defined in the next section.

Theorem 1. *Let $\mathcal{L}$ be a nonempty regular language, and let X be the associated sofic shift. Then*

$$h(\mathcal{L}) = \limsup_{n\to\infty} \frac{\log|\mathcal{W}_n(\mathcal{L})|}{n} = h_t(X).$$

Using Topological Entropy to Compute Distance

To use topological entropy to measure the complexity of $\mathcal{L}_1\Delta\mathcal{L}_2$, which is a regular language, we must first recast a DFA into a sofic shift that represents similar information. The common method to turn a DFA into a sofic shift is to simply forget the information regarding initial and final states. In that case, you can easily construct a single sofic shift that represents two very different regular languages [PYY19].

To deal with the issue of a single sofic shift representing different languages, we define another method to transform a DFA into a sofic shift. To explain this method, we first describe two additional operations that can be performed on DFA, trimming and computing the essential subgraph. The operation of *trimming* a DFA is to remove any state that is not included in some accepting path for that DFA. This operation does not change the language represented. A state q is removed during trimming if and only if q satisfies at least one of two properties: (1) there is no path from the initial state to q, or (2) there is no path from q to any final state. These conditions can be tested using a simple depth-first search. The second operation is computing the essential subgraph. Given a representation G of a sofic shift, the *essential subgraph* is the largest subgraph of G such that each vertex has at least one incoming edge and at least one outgoing edge. The essential subgraph of G can be calculated by iteratively deleting vertices that have no directed edge entering that vertex or no directed edge leaving that vertex.

Given a regular language $\mathcal{L}$, we define the *associated sofic shift* to be the sofic shift with representation G, where G is constructed from the DFA representing $\mathcal{L}$ by (1) trimming the DFA, (2) ignoring initial and final states, and (3) computing the essential subgraph.

As discussed previously, we believe that topological entropy is a good candidate for measuring the complexity of the symmetric difference between two regular languages (now that we can convert each regular language to a sofic shift), and here we use this concept to define a distance between regular languages. We call this distance function the entropy sum distance. It was inspired by first considering the topological entropy of the symmetric difference directly, i.e., $h_t(\mathcal{L}_1\Delta\mathcal{L}_2)$. However, since topological entropy only captures the irreducible component of highest complexity and does not account for the rest of the irreducible components, more information is gathered by computing the topological entropy of each summand of the symmetric difference directly and then summing the results. This leads us to the following definition: the *entropy sum distance* between two regular languages $\mathcal{L}_1$ and $\mathcal{L}_2$ is defined as

$$H_S(\mathcal{L}_1,\mathcal{L}_2) = h_t(\mathcal{L}_1 \cap \overline{\mathcal{L}_2}) + h_t(\overline{\mathcal{L}_1} \cap \mathcal{L}_2).$$

The entropy sum distance is a good candidate for measuring the distance between regular languages because, as we proved, it is a pseudometric and we give conditions on when it is granular (i.e., for any two points in the space you can find a point between them) [PYY19], and it can be efficiently computed. For more information on this distance function and its relationship with Jaccard-based distances, see [PYY19].

References

[BP97] Marie-Pierre Béal and Dominique Perrin, *Symbolic dynamics and finite automata*, Handbook of formal languages, Vol. 2, Springer, Berlin, 1997, pp. 463–505. MR1470015

[CDFI13] Cewei Cui, Zhe Dang, Thomas R. Fischer, and Oscar H. Ibarra, *Similarity in languages and programs*, Theoret. Comput. Sci. **498** (2013), 58–75, DOI 10.1016/j.tcs.2013.05.040. MR3083514

[CM58] Noam Chomsky and George A. Miller, *Finite state languages*, Information and Control **1** (1958), 91–112. MR108417

[HU79] John E. Hopcroft and Jeffrey D. Ullman, *Introduction to automata theory, languages, and computation*, Addison-Wesley Series in Computer Science, Addison-Wesley Publishing Co., Reading, Mass., 1979. MR645539

[LM95] Douglas Lind and Brian Marcus, *An introduction to symbolic dynamics and coding*, Cambridge University Press, Cambridge, 1995. MR1369092

[PYY17] Austin J. Parker, Kelly B. Yancey, and Matthew P. Yancey, *Regular language distance and entropy*, 42nd International Symposium on Mathematical Foundations of Computer Science, LIPIcs. Leibniz Int. Proc. Inform., vol. 83, Schloss Dagstuhl. Leibniz-Zent. Inform., Wadern, 2017, pp. Art. No. 3, 14. MR3755296

[PYY19] Austin J. Parker, Kelly B. Yancey, and Matthew P. Yancey, *Definitions and properties of entropy and distance for regular languages*, Dynamical systems and random processes, Contemp. Math., vol. 736, Amer. Math. Soc., Providence, RI, 2019, pp. 139–169, DOI 10.1090/conm/736/14845. MR4011910

[Rot81] Uriel G. Rothblum, *Expansions of sums of matrix powers*, SIAM Rev. **23** (1981), no. 2, 143–164, DOI 10.1137/1023036. MR618637

[Sal21] D. Saltman, *SCAMPing*, Notices of the AMS (2021).

[Yan19] K. Yancey, *A nonacademic career track and the balance it brings*, Notices of the AMS **66** (2019), no. 9, 1440–1442.

Kelly B. Yancey

Credits

Figures and author photo are courtesy of Kelly B. Yancey.