Some Logical Limits of E-Discovery

Lon A. Berk

Follow this and additional works at: https://scholar.smu.edu/scitech

Recommended Citation
Available at: https://scholar.smu.edu/scitech/vol12/iss1/2
Some Logical Limits of E-Discovery

Lon A. Berk*

ABSTRACT

Federal civil litigation is based upon the premise that the parties are able to collect and disclose relevant information. However, the huge growth of digital information in the custody of parties makes it increasingly difficult and burdensome for parties to comply with discovery obligations and to address the merits of their disputes. To satisfy discovery obligations when digital information is at issue, parties must use electronic methods of searching through data for relevant information. This paper discusses a theorem from computation theory – Rice's theorem – regarding the limits of what can be determined by algorithms, applies that theorem to search algorithms, and discusses the impact the theorem has on the ability of parties to evaluate and comply with discovery obligations.

I. INTRODUCTION

Justice Oliver Wendell Holmes, Jr. said, “[t]he life of the law has not been logic; it has been experience.” A naïve view of electronic discovery (“e-discovery”) ignores Holmes’ dictum. Perhaps understandably, some believe that e-discovery requires more of a need for logic than for experience. The prevalence of search engines, such as those used by Google, MapQuest or Amazon.com, might leave some practitioners with the impression that electronically stored information (“ESI”) is readily searchable with little effort or specialized training and that, in connection with e-discovery, experience can be replaced by logic. According to this view, all that is required is to identify key words, type them on the search bar for a suitable search engine, and press enter, with the result that all documents relevant to the claim or defense of any party are identified. One goal of this paper is to convince the reader that such a view of e-discovery is seriously misguided. The basic principles governing algorithms show that there is no “silver bullet,” no mechanical means, of searching ESI for relevant information. In other words, logic establishes bounds on what can be accomplished in the context of electronic discovery.

1. Thanks are due to Professors Charles Fried and Linda Wetzel and to my colleague John Woods for very helpful comments on this paper. Any mistakes are my own.

* Lon A. Berk is a partner at Hunton & Williams, LLP. His practice focuses on complex civil litigation with experience that includes the trial and appeal of cases involving commercial disputes in state and federal courts nationwide. Mr. Berk also regularly advises and represents clients in connection with liabilities arising out of emerging technologies, including issues relating to internet security and electronic discovery. He received his Ph.D. from the Massachusetts Institute of Technology and his J.D. from Harvard Law School.

The premise of civil litigation in the United States, at least theoretically, since the 1938 adoption of the Federal Rules of Civil Procedure, is that claims passing the threshold test of Rule 12 motion practice should enter a phase of information exchange, or discovery, where the parties request and exchange information relevant to the claims or defenses of any party. During this discovery phase, the parties should, subject to their rights to protect privileged and confidential information, exchange all information relevant to their claims and defenses, with minimal judicial intervention. The aim of discovery is to eliminate "trial by ambush" or surprise so that by the trial date, all parties have a solid understanding of the evidence supporting and disconfirming their litigation positions. Commentators and case law repeatedly make clear that the purpose of the discovery rules is to permit the parties to conduct a broad investigation of the facts that may assist in the presentation and preparation of a case for trial.3

The aim of discovery is to provide all parties with full access to the factual information underlying the transactions in dispute so that the parties can evaluate the strengths and weaknesses of their positions, negotiate a settlement if appropriate, and present a complete statement of their positions to the fact finder. Where a party fails to make a required disclosure, Rule 37 permits sanctions:

If a party fails to provide information or identify a witness as required by Rule 26(a) or (e), the party is not allowed to use that information or witness to supply evidence on a motion, at a hearing, or at a trial, unless the failure was substantially justified or is harmless. In addition to or instead of this sanction, the court, on motion and after giving an opportunity to be heard:

(A) may order payment of the reasonable expenses, including attorney's fees, caused by the failure;
(B) may inform the jury of the party's failure; and
(C) may impose other appropriate sanctions . . . .4

Consequently, the discovery phase presents litigants with an opportunity to collect and review all relevant information but also carries the risk of sanctions for failure to make full disclosure. If the system works properly, litigants will realize the opportunity to gather pertinent information while avoiding the risk of sanction. For this system to work, litigants must be able to: (1) search their records for relevant information; (2) produce relevant records to other litigants; (3) obtain the records of other litigants; and (4) review and digest all produced information to evaluate its impact on the parties' litigation positions. Furthermore, the parties must be able to do all of the above within the time prescribed.

The explosion of digital information calls into question whether such an extensive search and review is possible. Because of the increased volume of e-mails, instant messages, and other digital forms of communications, litigants are now obligated to review and/or produce collections of millions, or even billions, of documents. Manual review of such an extensive amount of information imposes excessive costs and burdens that can make it virtually impossible to satisfy disclosure obligations.

Because new digital technology has exponentially increased the amount of information that must now be reviewed for production, it is increasingly likely a producing party will overlook discoverable information, potentially subjecting that party to sanctions. Similarly, by increasing the amount of information that must be reviewed for trial preparation, it is more difficult to assimilate such information and digest it into a format that can be used to address the merits of the dispute. Thus, the information explosion has threatened the twin goals of civil discovery: (1) the exchange of relevant information so that (2) decisions may be rendered on the merits.

Questions regarding the efficiency of algorithms designed to search ESI, therefore, become critical to the premise of civil litigation. The only hope for accomplishing the goals of discovery is through the use of such algorithms. The Sedona Principle, No. 11, for example provides, "[a] responding party may satisfy its good faith obligation to . . . produce relevant electronically stored information by using electronic tools and processes, such as data sampling, searching, or the use of selection criteria, to identify data reasonably likely to contain relevant information." This paper will focus on the use of search algorithms to satisfy discovery obligations and to digest large amounts of information produced. By showing that it employed an appropriate search algorithm to search for relevant digital information, a party could demonstrate that it satisfied its disclosure obligations. Similarly, by using an appropriate search algorithm, a party can ensure that it has reviewed databases to obtain the information necessary to evaluate and present its case. A critical issue in electronic discovery, therefore, is whether the parties have used an appropriate search algorithm.

Unfortunately, as discussed below, no bright-line mechanical test exists for determining whether a search algorithm meets the most natural benchmarks for success. Basic logical principles governing algorithms create limitations on what they can accomplish, an issue of which litigators should be aware. In particular, the natural measures of the success rate for a search algorithm may not be determinable. This recognition may require a

6. See id. at 23.
change of legal culture, both in terms of the goals of litigants and the treatment of their obligations. Among other things, it suggests that there can be no perfect exchange of information and that, in fact, litigants may not be able to determine how close they have come to achieving such an exchange. Discovery obligations need to be understood in this context, recognizing that, even under the best of circumstances, it is not always possible to evaluate the degree to which there has been full and complete disclosure.

II. A Simple Description of Search Algorithms

To motivate this discussion, this paper will use a high-level description of algorithms in general and search algorithms in particular. An algorithm is “a well-defined computational procedure that takes some value, or set of values, as an input and produces some value, or set of values as an output.” It is a set of instructions that can be performed without choice by a computational system, whether human or machine, on an input.

This paper will describe algorithms using a “pseudo-code.” To take an example, the following describes an algorithm:

PRINT-SECOND-ITEM (D)
[An algorithm that on input D, a list of items, outputs the second item in D]
1. Let d = FIRST-ITEM (D)
2. Replace D with REST (d, D)
3. Replace d with FIRST-ITEM (D)
4. OUTPUT d

For instance, if D is the list: (1,2,3,4)

9. A wide variety of different models of computation have been developed. Each has proved to be equivalent in the sense that precisely the same set of problems is computable in each model. The precise model of computation used is immaterial to the current discussion. See generally Joseph R. Schoenfield, Recursion Theory (2001); Michael Sipser, Introduction to the Theory of Computation (1997).
10. Algorithms are often described in the literature through “pseudo-code,” a statement of instructions written in a mixture of English and an informal programming language, rather than in a programming code. Generally, the entries should be self-explanatory. Algorithms can use other algorithms as subroutines. For instance, FIRST-ITEM is an algorithm that returns the first item in a list; and REST (d, D) returns the list that appears after the first occurrence of d in a list D. For a description of a procedure in pseudo-code to be an “algorithm,” the subroutines must be algorithms as well. These subroutines, and the others used in this paper, meet that requirement. See generally Schoenfield, supra note 9; Sipser, supra note 9.
11. Lists are identified by enclosing list items in brackets. Thus, (1) needs to be distinguished from 1. The former is the list whose first and only item is the
and is input, this algorithm will output "2." First, the algorithm takes the list as input and at step one, let d be whatever is the first item of D. Second, the algorithm takes whatever occurs after the first item of D and replaces D with that. After the second step, D will be \( \langle 2,3,4 \rangle \), and d will be 1. In the third step the algorithm takes the first item of what is now D and replaces what is now d with that item. In step four, d is returned.

For some sorts of inputs, \( \text{PRINT-SECOND-ITEM} \) (D) will have no instruction to follow. For instance, if we input the number, 1,234, as opposed to the list of the numerals occurring in that number, or if we input a list with less than two entries, such as \( \langle 1 \rangle \), the machine has no instruction on what to do and, accordingly, yields no output.\(^{12}\)

Two different types of results might be obtained when an algorithm is run on an input. The algorithm can yield a defined result, or output, or it can be undefined, yielding no result. Where it is undefined, the algorithm does not halt. Where it yields a defined result, the algorithm halts.\(^{13}\) Where M is an algorithm that on input D yields output d, we will write:

\[
M(D) \Rightarrow d
\]

Thus, given the above, we have:

\[
\text{PRINT-SECOND-ITEM} \langle 1,2,3,4 \rangle \Rightarrow 2
\]

A search algorithm is an algorithm that performs a search over some database or collection of information. The idea is that the search algorithm will retrieve or permit the retrieval of all documents satisfying specified criteria. To simplify the terminology for purposes of this paper, assume that a database can be represented as a list and that a query can be represented as a list of words. While these assumptions ignore certain important aspects of search algorithms, these simplifications will not impact our conclusions and will make the discussion much easier.

A search algorithm will then be an algorithm with two inputs, both of which are lists. The first is a list of documents, and the second is a list of words. The output will be another list (or database), each element of which is a list that consists, first, of an identification of a document in the database that has been input and, second, a number, or weight, for that document. To simplify further, assume that the weights have been normalized so that they are on the interval between zero and one. Where M is a search algorithm

number 1; the latter is the number 1. \( \langle \rangle \) is an empty list; i.e. the list contains no items.

12. In such circumstances, one could have programmed the machine to output "error." That would be a different algorithm.

13. For technical reasons, assume that the algorithm's instructions are such that if a machine proceeds from line n to line m where m>n, then line m defines an action that the machine is able to perform. Otherwise, assume the machine simply enters a loop.
that, when given input $D$, a database, and $Q$, a query, and $DW$ is another database that lists weights for items in $D$, we write, using the above notation:

$$M(D,Q)\Rightarrow DW$$

And, if $d$ is an item in a database $D$, then:

$$M(D,Q)(d)\Rightarrow w$$

where $w$ is the weight such that $(d,w)$ is in $DW$. As this notation suggests, $M(D,Q)$, where $D$ and $Q$ are fixed, is itself an algorithm with input a document and output a number between zero and one. We specifically note this point here because it will become important later.

Using this notation to describe search algorithms suppresses some important issues, such as how to identify items in a database, how items in a database are ordered, what constitutes a word, and what constitutes a document. For a search algorithm to work as we have described, a structure must be imposed on the database that identifies and orders each item in the database. Further, this structure must be strong enough to allow its users to define documents and words. To see how difficult the task of imposing such a structure can be, imagine a library of words, not on pages, not ordered and not separated into books. How should one form a database for this library? Should one identify items in the database as words? What collections of words should one describe as pages? Once the pages are identified, what collections of pages belong together? And how should one order a collection of pages? In e-discovery, these issues may not always need resolution. For example, one may have a collection of e-mails that needs to be searched, which presumably can be ordered. Even a collection of e-mails, however, can present difficulties, and the structure imposed on a database may impact the results of a search algorithm.

To illustrate and motivate the following discussion, it is worth considering a very simple search algorithm as an example. This algorithm simply counts the words in a database and assigns a weight to the documents in that database by determining how many times the first item in a query appears in the document:

$$\text{SIMPLE-SEARCH} (D,Q)$$


15. This algorithm uses some other procedures: COUNT-ITEMS counts the items in a list; FIRST-ITEM returns the first item in a list; COUNT $(Q,d)$ returns the number of times an item $Q$ occurs in an item; ADD $(i,L)$ forms a new list by adding an item $i$ as the last element to a list $L$; REST $(d,D)$ returns the list that appears after the first occurrence of $d$ in a list $D$; and ID$(d)$ provides an identifier for $d$. 

Some Logical Limits of E-Discovery

[An algorithm that on input \( D \), a list of documents, and input \( Q \), a list of words, outputs a listing of weights for the documents in \( D \)]

1. Let answer = \( \{ \} \)
2. Let \( C = D \)
3. Let \( n = \text{COUNT-ITEMS}(D) \)
4. WHILE \( C \neq \{ \} \),
5. Let \( d = \text{FIRST-ITEM}(C) \)
6. Let \( m = \text{COUNT}(\text{FIRST-ITEM}(Q), d) \)
7. Add \( \langle ID(d), m/n \rangle \) to answer.
8. Replace \( C \) with \( \text{REST}(d, C) \)
9. END
10. RETURN answer

This algorithm, given a list of items (a database), creates a new list, each item of which identifies an item in the original database and assigns it a weight. The weight assigned will be the number of times the first item of the query, \( Q \), occurs in the identified document, divided by the total number of words that occur in all documents in the database. Thus, suppose we input into \( \text{SIMPLE-SEARCH} \) the query \( \langle \text{'Apple'} \rangle \) and the database, \( D \), consisting of the following three electronically stored documents:

- \( D_1 \): an apple a day keeps the doctor away
- \( D_2 \): an apple tree grows in the big apple
- \( D_3 \): a pear is better than a plum.

The algorithm will count the words occurring in \( D \). Seventeen words occur in \( D \). The algorithm will then output a database, \( D_W \), consisting of the following pairs:

- \( \langle D_1, 1/17 \rangle \)
- \( \langle D_2, 2/17 \rangle \)
- \( \langle D_3, 0 \rangle \)

In the notation above, \( \text{SIMPLE-SEARCH}(\langle D_1, D_2, D_3 \rangle, \langle \text{'apple'} \rangle) \Rightarrow \langle \langle D_1, 1/17 \rangle, \langle D_2, 2/17 \rangle, \langle D_3, 0 \rangle \rangle \).

\( \text{SIMPLE-SEARCH} \), like most of the algorithms lawyers are familiar with, is based upon a key-word search. Such algorithms are not the only kind that might be employed to search ESI. Keyword searches present sev-

---

16. Algorithms can assign weights to documents in ways other than using counts, and the simplicity of this example should not be over-generalized.

17. Search algorithms generally begin by creating an index of all terms that occur in the database with reference to the documents in the database. Thus, the first step in searching \( D \) is to create an index in which "apple" is listed as occurring in \( D_1, D_2 \) and \( D_3 \). This step saves not only computational resources, making it unnecessary to sort through the data repeatedly, but also allows the creation of complicated weighting systems. See, e.g., D.A. Grossman & Ophir Frieder, Information Retrieval: Algorithms and Heuristics, 182-95 (2d ed. 2004).
eral problems that derive from the ambiguities and vagueness of natural languages. For instance, when searching for documents concerning Apple computers, one would not want to treat D1, D2, or D3, which concern fruit, as having any relevance. Other problems with keyword searches arise from spelling errors, idiolects, and abbreviations. This aspect of SIMPLE-SEARCH is not, however, the focus of this paper. Other limitations on search algorithms are, perhaps, even more fundamental.18

One such issue is the manner by which items in the database are identified, which affects the results returned by SIMPLE-SEARCH. If, for instance, D1 and D2 were identified as a single item, D4, SIMPLE-SEARCH would return \( \langle \langle D4, 3/17 \rangle, \langle D3, 0 \rangle \rangle \). Or suppose the items in D were identified and ordered differently, say as:

- D1: an apple a day keeps the doctor away
- D2: an apple tree grows
- D3. in the big apple a pear is better than a plum.

In this case, SIMPLE-SEARCH would weight each item the same, returning:

\( \langle \langle D1, 1/17 \rangle, \langle D2, 1/17 \rangle, \langle D3, 1/17 \rangle \rangle \).

The determination of which items in a database should be treated as documents impacts the results of a search. Thus, the weights assigned by a search algorithm may be relative not only to the item that is weighted but also to how the items are broken up within the database where the item appears.

Further, the outcome of SIMPLE-SEARCH is a weighting of documents. That is to say, the algorithm searches and ranks the items relative to each other and does not assign absolute values. Some algorithms can be designed to provide absolute values, as opposed to relative weightings, but SIMPLE-SEARCH does not do so.

These problems with SIMPLE-SEARCH can run contrary to the goals of attorneys attempting to comply with civil discovery obligations. SIMPLE-SEARCH \((D,Q)\), therefore, is not the search algorithm that should be used in the discovery context. If one wanted to find all the times that Shakespeare used the name “Juliet,” for example, SIMPLE-SEARCH would be a satisfactory algorithm to use. Thus, determining which search algorithm is appropriate in a certain setting depends upon the purpose for which the algorithm will be used.

### III. Problems

In civil discovery, attorneys seek and are obligated to produce material that satisfies Rule 26. Subject to objections of burden, attorneys are entitled

---

to obtain and required to produce, "any nonprivileged matter that is relevant to any party's claim or defense . . . ."19 Relevant information or evidence is, in turn, defined as "evidence having any tendency to make the existence of any fact that is of consequence to the determination of the action more probable or less probable than it would be without the evidence."20 In civil discovery, therefore, the test of whether a document is within the scope of discovery is not a question of degree. If the item has any tendency to make the existence of a fact of consequence to a party's claim or defense more or less probable, it is discoverable. Although one document may be more relevant than another, it meets the test of discoverability if it is relevant, irrespective of whether other, more relevant documents exist.

This issue should not be confused with the question of burden. The discovery rules provide that a party from whom discovery is sought may seek a protective order and that "[t]he court may, for good cause, issue an order to protect a party or person from annoyance, embarrassment, oppression, or undue burden or expense," restricting discovery.21 The question of undue burden or expense, therefore, is a relative one, but the question of whether the item is relevant is not.

This suggests that a search algorithm to find documents within the scope of civil discovery should assign absolute, not relative weights. To satisfy this requirement, the algorithm should assign one of two weights to documents: 0 or 1.22 That is, a search algorithm, M, satisfactory for civil discovery, should satisfy the following:

\[(R1) \text{(Boolean valued)} \quad \text{For all items}, \quad d, \quad \text{for all databases}, \quad D, \quad \text{and} \quad \text{for all queries}, \quad Q, \quad M(D,Q)(d) = 0 \text{ or } M(D,Q)(d) = 1.\]

Alone, this requirement is not difficult to meet. Many search algorithms satisfy (R1). All one needs to do to obtain a search algorithm that satisfies (R1) is to assign the documents with weights greater than or equal to a certain number the value 1 and to assign those with less than that number the value 0. For example, we can modify SIMPLE-SEARCH to:

\[\text{BV-SIMPLE-SEARCH} \quad (D,Q,r)\]

\[\text{[An algorithm that on input } D, \text{ a list of documents, and input } Q, \text{ a list of words, outputs a listing of weights for the documents in } D]\]

22. 0 means that the document fails the search criterion; 1 means that the document satisfies the search criterion.
23. This algorithm uses some other procedures: COUNT-ITEMS counts the items in a list; FIRST-ITEM returns the first item in a list; COUNT (Q,d) returns the number of times an item Q occurs in an item; ADD (i,L) forms a new list by adding an item i as the last element to a list L; and REST (d,D) returns the list that appears after the first occurrence of d in a list D.
1. Let answer = \{ \}
2. Let C=D
3. Let n = COUNT-ITEMS (D)
4. WHILE C ≠ \{ \},
5. Let d = FIRST-ITEM (C)
6. Let m = COUNT (FIRST-ITEM(Q),d)
7. IF m/n=r
8. ADD (ID(d), 1) to answer.
9. OTHERWISE ADD (ID(d), 0) to answer.
10. Replace C with REST(d,C)
11. END
12. RETURN answer

BV-SIMPLE-SEARCH will weight documents in a database as 1 if SIMPLE-SEARCH gives them a weight greater than or equal to r. Correspondingly, BV-SIMPLE-SEARCH will weight documents as 0 if SIMPLE-SEARCH gives them a weight less than r. The determination of what value of r should be used for a particular search would be a matter of judgment relative to query, the size of the database, the range of weights assigned, and other factors. It is relatively easy, therefore, to modify a search algorithm so that (R1) is met.

(R1) is not the only requirement, however, that a satisfactory search algorithm must meet. For example, BV-SIMPLE-SEARCH (D,Q,0) will be very broad, assigning the value 1 to an item whether that item contains only one or 1,000 occurrences of the first item of the query. Where a database contains millions of documents, litigants may need a more restrictive search algorithm, i.e., one that more successfully culls the relevant from the non-relevant documents in the database.

IV. SUCCESS OF A SEARCH ALGORITHM

The goal of a search is to collect all and only those documents that are relevant, and search algorithms satisfying (R1) can be unsuccessful in this sense. Some may assign too many documents the value 1; others may assign too few documents that value. They can be over-or under-inclusive. Accordingly, the two measures of a search algorithm’s success are recall and precision. Recall is the percentage of relevant documents obtained by the search. Where \(R_0\) is the number of relevant documents in the database to be searched, and \(S\) is the number of documents assigned the value 1 by the search, \(S/R_0\) is the recall of the search. For example, if there are 4,000 documents meeting the Rule 26 test in a database of 40,000 electronic documents, and a search algorithm returns the value 1 for 3,800 of the relevant documents, the recall of the search algorithm will be 95%.

Obviously, good recall is one measure of a successful search algorithm, but it is clearly not the only one. A search that assigns the value 1 to all 40,000 documents in the data will, for example, have perfect recall – all 4,000 relevant documents will be produced – but such a search will be useless. Accordingly, the second measure of a search’s success is precision.
Roughly, precision is the percentage of relevant documents that are contained within the set of all documents selected by the algorithm. That is, where $S$ is the number of documents assigned the value 1, and $R_s$ is the number of documents meeting the Rule 26 test among those that are selected by the search, precision is $R_s / S$. For example, a search algorithm applied to 40,000 electronic documents that returns the value 1 for 5,000 documents, out of which 3,800 are within Rule 26, has a 76% precision rate.

Intuitively, the goal of any search is to have both precision and recall as close to 100% as possible. With recall and precision both at 100%, the algorithm will collect all and only the relevant documents from a data set. Thus, if given a database $D$, one could design a query $q$ and search algorithm $M$ such that $M(D,q)$ has 100% recall and precision. A litigant could then use $M(D,q)$ to collect only the relevant documents in $D$ and nothing more.

A 2008 federal district court case, Victor Stanley, Inc. v. Creative Pipe, Inc., demonstrated the need for a search algorithm with high precision. In that case, the defendant had produced approximately 165 electronic documents that it claimed were protected as work-product and by the attorney-client privilege. The court ordered the parties’ computer forensic experts to meet in order to develop a joint protocol to search and retrieve relevant ESI responsive to the plaintiff’s document requests. The agreed-upon search protocol included keyword/phrase search terms that were aimed at locating responsive ESI. The developed protocol resulted in the production of some privileged documents. A second search was then applied to the documents to identify privileged documents. The ESI ultimately produced to the plaintiff contained approximately 165 privileged documents. Evidently, the precision of the second search algorithm was less than 100%, the result being that not all privileged documents were collected. The producing party had, unfortunately, not entered into a clawback agreement, and accordingly the court determined that the producing party had waived the privilege by producing the privileged documents.

On the other hand, the need for an algorithm with high recall is demonstrated by In re Seroquel Products Liability Litigation. There, the court imposed discovery sanctions because, among other things, a keyword search failed to identify relevant e-mails and documents. The producing party did

25. Id. at 253-54.
26. Id. at 254.
27. Id. The court found it significant that the protocol was developed to identify only responsive documents, while the protocol did not identify documents that were either protected by the privilege or were attorney work-product.
28. Id. at 255, 262.
30. Id. at 662, 665.
not consult with plaintiffs regarding the design of a word search and conducted a search that omitted generic names, acronyms, British spellings, and common misspellings and did not account for spacing or hyphenation. The court found that the discovery word search was "plainly inadequate" and sanctioned the producing party for failing to produce documents that had not been identified by the search.

These two decisions demonstrate the risk of using search algorithms with poor recall and precision measurements. Thus, the challenge for a practitioner is to choose a search algorithm with as close to perfect recall and precision as possible. In doing so, counsel of the producing party will ensure that his client has produced all ESI that meets the Rule 26 test, but nothing more, while opposing counsel will ensure that her client has obtained all the information she is entitled to receive. Where a producing party fails to use such a search algorithm, it has failed to comply with its obligations under the Federal Rules of Civil Procedure.

The problem then confronting litigants is how to choose a Boolean-valued search algorithm – that is, one satisfying (R1) – and a query, Q, that will search a database, D, with recall and precision meeting specified benchmarks. Obviously, the goal is to meet a 100% benchmark, but such a high goal is likely unattainable. Suppose that practitioners and courts accept that discovery obligations are met so long as the search algorithm, M, searches a database, D, and query, Q, with recall at least m and precision at least n. In this situation, a practitioner would meet her discovery obligations if the recall of M(D,Q) is greater than or equal to m and its precision is greater than or equal to n.

How are courts and practitioners to ensure that the parties used such an algorithm without actually searching the database for relevant documents and comparing the results of the algorithm? The most obvious solution is to use an algorithm that takes as inputs search algorithms, databases, queries, and outputs 1 if the benchmark is met and 0 if it is not. Using such a method, a practitioner would know whether the method used to collect relevant ESI met the specified benchmarks. Thus, the goal is to have an algorithm, P, such that

\[ P(M(D,Q)) = \begin{cases} 1, & \text{if the recall of } M(D,Q) \text{ is greater than or equal to } m \text{ and its precision is greater than or equal to } n; \\ 0, & \text{otherwise} \end{cases} \]

31. *Id.* at 661 n.7.
32. *Id.* at 662.
33. FED. R. CIV. P. 26(a)(1), (3).
Unfortunately, logic guarantees that this result is unattainable. A well-known theorem from computation theory, Rice’s Theorem, demonstrates that no algorithm, such as P, exists if some of the documents in D are relevant and some are not. More specifically, where some of the documents in a database are relevant for production purposes and some are not, it is impossible to mechanically determine whether a search algorithm meets (R1) and a query meets the specified recall and precision benchmarks. Perfection need not be the goal. Regardless of the values chosen for m (recall) and n (precision), no mechanical test – or algorithm – can be used to decide whether a search algorithm on a database and query has recall value m and precision value n.

While the proof of Rice’s Theorem is relatively easy, its implications for e-discovery are profound. If no mechanical test can determine whether a non-trivial algorithm will identify relevant documents in a database with specified recall and precision, then no test will be able to determine whether the search algorithm used to obtain such documents from ESI meets that standard. No matter how much one tries, she cannot ensure that the search algorithm employed will identify all, and only, documents meeting the Rule 26 test with a specified recall and precision. A fixed standard is impossible, therefore, and a practitioner may be unable to demonstrate that the search technique used was the best or even the most appropriate.

This result underscores the wisdom of the Sedona principles: the idea that the design of search algorithms and queries should be a matter of cooperation among the parties. Only where opposing parties cooperate can such parties assure that failure to identify and produce relevant documents will not subject the producing party to sanctions because the producing party failed to design an appropriate query and search algorithm. If the requesting party and the producing party agree on the algorithm and query to be employed to search ESI, then the parties can avoid motions for sanctions based upon the failure to employ a better algorithm and query.

V. Rice’s Theorem

Rice’s Theorem states that no algorithm can be used to decide whether an algorithm satisfies a non-trivial extensional property. One can unpack the meaning of this statement without much trouble. First, note that an algorithm (M) decides whether or not an input of a certain sort has a property (P) if the following holds for every input (d) of that sort:

\[ M(d) = \begin{cases} 1, & \text{if } d \text{ has the property } P; \\ 0, & \text{if } d \text{ does not have the property } P. \end{cases} \]

Second, since algorithms are no more than sets of rules, they can be coded with names and stored in electronic databases so that the algorithms themselves are the subjects of search algorithms. This is not surprising because modern computers would be impossible without a database of algorithms. Further, these databases are potential subjects of litigation. For example, an intellectual property case may involve questions about which
algorithms were obtained or created by a party on what date. Also, a lawsuit regarding whether a party has properly executed its e-discovery obligations may turn upon which search algorithms the party possessed in its own database. Thus, algorithms themselves can be ESI, and we may apply search algorithms to databases of algorithms.

A third point of interest is the notion of "extensional property." For example, given any algorithm (M), let us use $R_M$ to refer to the set of inputs to which M assigns the value 1, and let us say that $R_M$ is the set of inputs accepted by M. Different algorithms may of course accept the same inputs. In other words, there are circumstances under which $R_M = R_N$ even though M and N are different algorithms.

An extensional property, $P$, is a property that, if had by an algorithm, $A$, is had by every algorithm that accepts the same set of inputs as $A$. More formally, let M and N be two algorithms, and suppose that $R_M = R_N$. Then, an extensional property is had by M if and only if that property is had by N.

For the purposes of this paper, the notion of an extensional property is important because, where search algorithms satisfy (R1), the property had by all algorithms having recall and precision meeting specified benchmark values on a given database and given queries is an extensional property. That is, let us suppose that we have a fixed database, D, and two queries, $Q$ and $Q'$. Then, if M and N are search algorithms satisfying (R1), as noted above, $M(D,Q)$ and $N(D,Q')$ will be algorithms with inputs documents and outputs either 0 or 1. Further, if $M(D,Q)$ assigns the same documents the value 1 as does $N(D,Q')$, then M using $Q$ and N using $Q'$ will necessarily have the same recall and precision on D. So, if $M(D,Q)$ meets the specified benchmarks for recall and precision, so will $N(D,Q')$.

Now, the significance of Rice's Theorem becomes evident. The upshot of the theorem is that no algorithm exists for distinguishing among non-trivial satisfactory search algorithms based upon extensional properties. Additionally, since meeting specified recall and precision benchmarks on a fixed database is an extensional property, by Rice's Theorem, no algorithm can test whether a party has employed a search algorithm satisfying (R1) that met the specified benchmark.

The proof of Rice's Theorem is relatively straightforward.35 It derives from the "halting problem," which was uncovered in the 1930s. The halting problem is the problem of establishing that an algorithm halts on an input. It can be shown that no algorithm decides the halting problem.36 From the halting problem, it is only a short step to the version of Rice's Theorem of

35. Those who are familiar with, or do not wish to review, the proof can skip to the beginning of the next section.

36. To see this, begin with an algorithm, $H$. Then, where N is any algorithm and $d$ is any input,

$$H(N, d) \Rightarrow 1$$

if and only if N halts on input $d$; and
interest in this paper. Suppose we have an algorithm, P, which decides whether, given a database D containing some relevant and non-relevant documents, a search algorithm and a query meet the benchmark recall and precision values. There will be some search algorithm SEARCH, for which P[SEARCH(D,Q)]=1.37

Now, define the following algorithm:

RICE-SEARCH (M,w)
1. LET C be the following algorithm:
2. On input d run M on w.
3. Next run SEARCH(D,Q) on d.
4. RETURN 1 if SEARCH(D,Q)(d)==1.
5. IF P(C) \(\Rightarrow\) 1, RETURN 1.
6. IF P(C) \(\Rightarrow\) 0, RETURN 0.

This algorithm returns 1 if M halts on input w and returns 0 if M does not halt on input w. RICE-SEARCH (M,w) works as follows: first, on inputs M and w, RICE-SEARCH constructs an algorithm, C. In turn, the algorithm C works as follows: on input d, C first runs M on w and then runs SEARCH(D,Q) on d. After having constructed the algorithm C, RICE-SEARCH then runs P on C and returns as output whatever is returned by P.

Now, the only way the algorithm C would get to line 3 is if at line 2 M halts on w. If M does not halt on w, then C will never get to line 3 and so will not halt either. And, if M halts on w, SEARCH(D,Q) will be run on

\[ H(N, d) \Rightarrow 0, \text{ if } N \text{ does not halt on input } d. \]

Now, ask whether a given machine halts when given itself as an input, which is not unusual. In fact, many algorithms succeed only because they take themselves as one of their inputs. Thus, the question is whether N halts on input N. Obviously, if one had H as above, she could apply H to (N, N) and obtain the answer. So from H, one can construct a new algorithm as follows:

DIAGONAL (N)
1. IF H(N,N)\(\Rightarrow\)0
2. RETURN 1.
3. OTHERWISE, GOTO line 1.

But now, what about DIAGONAL? Does DIAGONAL halt on input DIAGONAL? If it does, then H(DIAGONAL, DIAGONAL)\(\Rightarrow\)1, so DIAGONAL(DIAGONAL) does not halt. That is, H(DIAGONAL, DIAGONAL)\(\Rightarrow\)0, which means DIAGONAL(DIAGONAL) does halt. Thus, we have a contradiction. It follows that, because DIAGONAL was constructed only from the assumption that an algorithm, H, existed that decides whether an algorithm halts on an input, there can be no algorithm H.

37. One may not know what that algorithm is, but there must be one. For instance, given a set of relevant documents, one can simply list them and construct an algorithm based upon that list. So, at least one search algorithm will satisfy the assumption.
input \( d \). So, in the case where \( M \) halts, \( C \) will output on \( d \) precisely what \( \text{SEARCH}(D,Q) \) outputs on \( d \). It follows that, if \( M \) halts on \( w \), \( C \) will accept the same inputs as are accepted by \( \text{SEARCH}(D,Q) \). So, if \( M \) halts on \( w \), then the algorithm \( C \) will have the same recall and precision as \( \text{SEARCH}(D,Q) \). And so, like \( P[\text{SEARCH}(D,Q)] \), \( P(C) \) will return 1. It follows that, if \( M \) halts on \( w \), \( \text{RICE-SEARCH}(M,w) \) returns 1.

On the other hand, if \( M \) does not halt on \( w \), \( C \) will never halt on any input and, therefore, will never accept any input. In that circumstance, \( C \) will not have the specified recall and precision. And so \( P(C) = 0 \). It follows that \( \text{RICE-SEARCH} (M,w) \) returns 0 when \( M \) does not halt on \( w \). But then:

- \( \text{RICE-SEARCH} (M,w) \Rightarrow 1 \), if \( M \) halts on input \( w \); and
- \( \text{RICE-SEARCH} (M,w) \Rightarrow 0 \), if \( M \) does not halt on \( w \),

and thus, \( \text{RICE-SEARCH} \) decides the halting problem.

However, no algorithm is capable of deciding the halting problem. Because all that was needed to construct \( \text{RICE-SEARCH} \) was \( P \), it follows that \( P \) does not exist, and no algorithm decides whether a search algorithm on a query and database meets the specified benchmark recall and precision values.

### VI. Conclusion

Given the goals of discovery, litigants should strive to maximize the recall and precision of the search algorithms they use. The most natural benchmarks to use to determine whether litigants have complied with their discovery obligations are in terms of the recall and precision of the search algorithms. Rice’s Theorem, however, provides that no algorithmic test can determine whether a given database, search algorithm, and query meet benchmark recall and precision values. Moreover, based upon Rice’s Theorem, no method exists to ascertain whether a search algorithm meets such benchmarks absent a review of the entire database. For all we know, an unacceptable percentage of relevant documents were missed, or an unacceptable percentage of non-relevant documents were selected.\(^{38}\) Whatever benchmark is used, if it is in terms of a specified recall and precision, no algorithm can determine whether that benchmark is met.\(^{39}\)

---

\(^{38}\) One might propose that mechanical tests other than algorithms be used. This proposal, however, runs afoul of Church’s Thesis. To date, all mechanical methods that have been devised have turned out to be equivalent to what we have defined to be an algorithm. See generally Joseph R. Schoenfield, Recursion Theory (2001).

\(^{39}\) Jason Baron argues in favor of the use of benchmarks to study what search algorithms are appropriate for use in federal civil discovery. It is not clear what sort of benchmarks he has in mind, and the conclusions of his insightful article are consistent with the conclusions here, unless these benchmarks are extensional properties for which there is a mechanical (algorithmic) test. See Baron, supra note 18, at 244-45.
This conclusion arises whether or not we have worked out the semantics of words sufficiently to ensure that queries seek relevant material—whether, for example, one has figured out a method for distinguishing whether a search for ‘bank’ yields documents concerning a financial institution or a river’s edge. Rice’s Theorem is independent of concerns about vagueness and ambiguity. It imposes a limitation on how much we can mechanically determine regarding the success of search algorithms.

It follows that, at some point, judgment about electronic discovery must intervene over bright-line, mechanical tests. No proper mechanical test exists to evaluate whether a search algorithm has satisfactory recall and precision for use in discovery. This result underscores the importance of designing searches used in e-discovery in a cooperative manner with the consent of all parties.40 This conclusion also confirms that Holmes’s dictum that experience prevails over logic also applies in the context of e-discovery.

40. See White v. Graceland Coll. Ctr. for Prof’l Dev. & Lifelong Learning, Inc., No. 07-2319-CM, 2008 WL 3271924, at *11 (D. Kan. Aug. 7, 2008) (“While not all disputes regarding discovery of ESI can be prevented by early efforts by counsel to investigate and consider the possible forms discovery may be produced, many disputes could be managed and avoided altogether by discussing the issue before requests for production are served.”).