Identifying and Attributing Similar Traces with Greatest Common Factor Analysis

Fred Cohen
Fred Cohen & Associates
http://all.net/

ABSTRACT

This paper presents an algorithm for comparing large numbers of traces to each other and identifying and presenting groups of traces with similar features. It is applied to forensic analysis in which groups of similar traces are automatically identified and presented so that attribution and other related claims may be asserted, and independently confirmed or refuted. The approach of this paper is to identify an approximate algorithm that will find a large subset of greatest common factor similar groups of arbitrary factors in far less time and space than an exact algorithm using examiner-provided selection criteria for factor definition.

Keywords: similarity analysis; digital forensics; greatest common factor; attribution

I. INTRODUCTION AND BACKGROUND

Locard's Exchange Principle states that whenever two objects come into contact, a transfer of material will occur [1], and it follows that the traces of material left on each of those objects is evidence of that transfer and the presence of the two objects at the same place at the same time. Unlike physical evidence, digital forensic evidence (DFE) is not transfer evidence, in that it is not the result of contact between physical objects, but it is trace evidence, in that it consists of traces (i.e., sequences of bits) left by digital systems as a result of the processes that they undertake in their operations. [2] Notionally, similar mechanisms leave similar traces under similar inputs, subject to the inherent discontinuity of digital space.

DFE is also latent evidence in that it can only be observed by people through the use of tools, [2] and thus it follows that tools are required in order to analyze and view that evidence. As the volume of DFE increases, it becomes infeasible to examine it by purely human cognitive processes acting on simple representations, and thoroughness in the sense of examining traces to definitively determine what took place because the number of comparisons grows more than factorially with the length of the trace and there are a large number of possible event sequences that can lead to any given trace in the number of possible FSM states or instructions and data, and time available for trace production.[3] For this reason, in almost all cases, examination is limited to identifying consistency and inconsistency of traces, in the sense first identified in [4], in particular, identifying

internal and external consistency of traces, respectively, to each other, and "events". [2][3][5]

Unlike physical evidence, an exact duplicate of DFE can normally be made for examination purposes without altering or destroying original evidence. This is not to say that physical evidence such as the original media upon which DFE was first found has these properties, but rather, DFE is distinguished from the media upon which it exists and was produced within in that it has these and other specific properties. [2] These properties limit the interpretations of DFE and aide in identifying inconsistencies. Among these properties are that specific computational mechanisms produce the same output for the same input and initial state.

1.1 Attribution and similarity assumptions

The attribution problem has long been considered fundamental to information protection in a wide variety of ways, if only because without being able to attribute actions to actors, there is no way to induce consequences on the specific actor. In the more general sense, this can be thought of as the problem of establishing causality. [2] In the digital world, because we know that specific finite state machines [FSMs] produce the same output for the same input and state, we can drive hypothesized inputs and states to outputs and, with relative ease, reproducibly show consistency or inconsistency of hypotheses to the available traces (the forward direction), but the reverse direction yields the potential for exponential expansion of possible inputs, states, and FSMs (i.e., the digital space converges in the forward direction and diverges in the reverse direction). [2] Thus there are fundamental limits on attribution inherent in the limits of establishing causality.

As a result, an underlying assumption of similarity analysis for the purposes of attribution in general, and the methods herein in particular, is that the mechanisms used to produce "similar" traces stem from "similar" or identical mechanisms and inputs. We will call this the "same effect, same cause" (E⇒C) assumption, and it clearly refutable in the case of forgery. Thus, in the establishment of a causal chain from a hypothesized cause to an effect in the form of the available traces, a series of cause-effect relationships are hypothesized, and for the DFE portions of this chain, tests for consistency and inconsistency with hypothesized event chains are devised. If the tests demonstrate inconsistency, then the hypothesis cannot be correct, but because of the complexity issues previously identified, an attainable number of confirmations are, essentially never, definitive, while a single refutation is. While this is closely related to the philosophy of science results of Popper, [6] it is not identical, and care should be taken in such characterization. Perhaps the most famous quote relating to this is the logical fallacy "cum hoc ergo propter hoc" (roughly "with this therefore because of this"), which has been largely replaced by "correlation is not causality" with the advent of statistics. For more in-depth coverage of this issue, see chapter 7 of [2].

The E⇒C assumption is clearly wrong in certain circumstances. In particular, in the presence of intentional forgery, where traces are in the possession and/or control of any particular party, that party has the potential to produce fabrications that meet consistency requirements for going undetected. Additional care in dealing with chain-of-custody, data providence, and scientific methodology issues are thus needed, which is the reason that such issues are fundamental to the admissibility of evidence and testimony.[7][8][9]

1.2 Similarity and attribution

A great deal of research has been done in searching large collections of content for strings or other similar expressions, including searches for n-tuples and proximity. For example, searching large portions of the Internet for small sequences of words is now commonly done by millions of people on a daily basis using services like Google and other similar sorts of search engines. These issues have been studied throughout the history of computing, and have been the subject of well known works. [10][11] These techniques include providing metrics for similarity and sorting of search results based on those metrics for the purposes of presentation.

Many digital forensics tools provide search capabilities for looking for regular expressions within content, and some of these tools include the capability to index content in many forms, formats, and representations. The same is true of search engines designed as built-in or add-on operating system mechanisms, such as the Apple Spotlight mechanism.

There are also techniques such as stylometrics,[12] like writing or coding styles, and the use of pre-existing code collections from known locations such as Web sites, news groups, books, and less widely available sources, graphic design style, vocabulary, sentence structure, word usage, etc. Pedersen has been a leader in this sort of research, [13] and there are many examples of methods he has applied to analyze linguistic patterns for applications ranging from disambiguation of word sense in human sentences to detecting plagiarism.

While these approaches appear to be good ideas, and some of them have been significantly explored over the years,[14] from a DFE attribution standpoint, there is little definitive information that can be used to associate reliability information with them. Some such methods have been admitted in US courts for limited purposes,[12] but they have not, as a class, been well tested, or survived significant test cases. These methods generally surround the attempt to attribute metrics to known samples of individual behaviors and then detect the presence of similar values for those metrics within collections of traces associated with relatively small (on the order of a few hundred) known individuals to identify traces that are potentially attributable to the identified individual. They usually rely on N-grams of some sort, where they selection of the symbol set and sequencing criteria are defined by a syntax and N-grams within a linguistic syntax analysis. This is sometimes couched in a limited form in terms of "near", "next-

to", "before", "with", and other similar search modifiers. This approach returns to the problems of identifying potential causal mechanisms, the E⇒C assumption, and the problems of symbol set selection that lead to factorial time and space. [3]

1.3 The problem of group identification

Similarity of groups of traces performed so as to identify what groupings of traces are present and measure the extent to which they are similar, has not apparently been explored in any significant way. While there are some mathematical problems related to cliques[15] and a wide range of other similar sorts of things, to date, these have failed to address the challenges of digital forensics in terms of the need to identify groups of traces with content containing similar characteristics and features. In forensics cases, while searches for known or suspected content are often used, it is also quite common to have a corpus of traces (e.g., files, messages, database entries, or other structured or unstructured content) for which identifying similar or related groups of traces becomes a key issue in addressing legal issues, particularly the issues related to attribution.

A typical example is a case in which attribution of "forged" USENET postings to real authors, systems, or mechanisms is of import [16] (this matter will be used below as a case study). Other examples include, without limit, cases where similarity of authorship, sourcing, or delivery mechanisms is probative; cases where evolved versions of similar coded content, such evolutionary viruses or copyright infringement matters; cases involving alterations, such as image files created or edited with the same version of the same software package; cases involving metadata where files with similar metadata may be related; cases involving log entries where similar sequences of events may be found; and cases in which common authorship based on writing "style", word usage, typing errors, spelling errors, grammatical constructs, etc. are to be identified.

Another general area of potential applicability is in intrusion, anomaly, or behavioral detection and analysis; where groupings based on identified characteristics may be used to associated large numbers of items of interest with each other.

In such cases, algorithmic complexity of $|T|2$, where T is the set of traces being considered and $|T|$ is the number of traces being compared to each other, seems almost inevitable, simply because, at least notionally, each trace $t \in T$ must be compared to each other trace in some way. While this produces feasible solutions for cases with relatively small $|T|$, as $|T|$ grows, $|T|2$ grows far faster. In cases involving 106 traces, $|T|2=1012$, which is near or over the edge of available time and space in typical legal matters. Cases in the legal system have already involved examination of almost 106 traces, in the form of electronic mail messages, far greater volumes are common in network analysis, and a typical file system today has millions of files.

1.4 Considering similarity in terms of differences

One approach to understanding the limits of similarity is to understand the limits of technology related to differences. While comparing two files is feasible with a program like the Unix "diff" and "cmp" utility programs[17], finding optimal (i.e., minimal change) differences (arguably the opposite of similarities) between two files, also known as the minimum edit distance problem, is at least NP-hard,[18] and is a superset of the longest common sequence problem, which is at least complexity O(l2), where l is the length of the traces.[19] Thus, even finding optimal differences between pairs of traces is far too complex for everyday use.

Finding "nearly identical" traces can be done far more quickly, depending on the definition of "nearly identical". For example, for traces of similar length (i.e., the same number of lines), a sort of the traces O(|T|log(|T|)) and "diff" comparison in sequence may be used to identify nearly identical sequences, but for instances where the beginnings of the traces are where the differences lie, this will not yield useful results.

Pairwise difference comparison yields a minimum of |T|2 time, and even if a pair of traces are found to be related to each other with a measurable degree of similarity (e.g., the inverse of the metric used to measure differences) through such a method, the factors that cause each to be similar to (different than) others may be different from the factors that cause all others to be similar to (different than) others. Thus groupings of more than pairs are not produced by this approach.

1.5 Verifiability and presentability of results

In order to be useful in a legal situation, results of analysis should also be meaningfully presented in terms of the specific basis for claims and presentable so as to demonstrate those bases.

Some of the techniques used for similarity analysis, like the support vector machine (SVM) approaches, use a learning algorithm to form parameters that don't directly relate to typical human meaningful sequences.[14] The resulting similarity metrics seem to be meaningful, but to date, they have not been demonstrated so from a standpoint of identifying mechanisms for cause and effect. In addition, these approaches have only been applied with statistically meaningful results in cases where known good sample traces (i.e., no attempts to subvert the mechanism or forge anything) of all parties from a small corpus of parties (on the order of a few score) are available to test against a suspect trace that is also assumes to not be intentionally altered.

Such mechanisms also do not produce independently verifiable results, in the sense that, without access to the mechanism used to derive the results, they cannot be tested. Thus we must trust the mechanism to properly perform its function, which leads back to the problems of proof of program correctness, specification issues, and all of the other challenges of trusted computing. It would be preferable

to have mechanisms that allow and independent third party to examine the source content and the claimed results with minimal tools (e.g., a viewer of some sort) to confirm, on a case-by-case basis, that specific results are as claimed.

For example, the result of the "diff" program that shows the differences between two sequences can be demonstrated by making the changes to one sequence and verifying that it produces the other sequence. This can be done manually for small numbers of differences (highly similar sequences) or with a minimum of automation using a text editor and macro processor for sequence pairs with a larger number of differences. In either case, the reproduction and verification of results can be done with independent software on a different system and with limited effort.

These two criteria, independent verifiability of results with limited effort, and presentation of results in a manner that allows them to be verified and shown to independent reviewers with limited technical understanding (i.e., the triers of fact in a legal matter) are vital to success in the legal system.

2. THE GREATEST COMMON FACTOR APPROACH TO GROUP IDENTIFICATION AND CHARACTERIZATION

Prime numbers (primes) are a number theory concept defined as positive integers such that no other positive integer divides them to produce an integer. All positive integers can then be uniquely represented as the product of primes, and we talk about the prime factors of an identified integer as the set of primes that, when multiplied, produce that identified integer. The greatest common factor of two identified integers is the largest integer such that it divides both identified integers to produce integers. The greatest common factor (GCF) of two integers can be produced by forming the set of factors common to both integers. Thus, the prime factors of 72 are{2, 2, 2, 3, 3}, the prime factors of 48 are {2, 2, 2, 2, 3}, and the GCF of 72 and 48 is the product of {2, 2, 2, 3}, or 24. Conceptually, the greatest common factor of an identified set of numbers can then be determined by factoring all elements of the set and combining their respective sets of factors to form the greatest set of factors common to all of the identified set. Thus if we add 18, whose factors are {2, 3, 3} to the set {72, 48}, the GCF of the resulting set {72, 48, 18} is the product of {2, 3}, or 6.

2.1. Identifying GCF groups of similar integers

Consider, then the question of identifying "similar" positive integers from a set of identified integers. Given that all integers are the products of primes, we could identify the prime factors as a basis for similarity, and form groups of integers with sets of prime factors as similar to each other. Identical integers have identical prime factor sets, while integers that are not similar, even though they may be very close in numerical value, have very different factor sets. For example, {72, 48, 18, 7} has the following similar groups and their respective greatest common factors.

{72, 48} has GCF {2, 2, 2, 3}

{48, 18} has GCF {2, 3}

{72, 18} has GCF {2, 3, 3}

{72, 48, 18} has GCF {2, 3}

{7, anything other than 7} has GCF {1}

We can then group these by GCF sets and identify that there are 4 different groups of similar integers in this set. In particular, they are {72, 48}, {72, 18}, {72, 48, 18}, and {7}. This similarity result eliminates all of the other subsets because they either produced the singleton GCF {1} indicative of no similarity, or there was a larger set with the same GCF (in the case of {48, 18}, which was subsumed by the larger set {72, 48, 18}). We will call these largest sets GCF-similar groups of positive integers.

We note that the selection of number theoretical factors as the basis for defining similarity was arbitrary in this example. We could just as well have selected the number of even vs. odd digits, the length of the integers, the number of straight line segments vs. curved line segments in the depiction of the digits, the number of angles contained within the depiction of the integer, the number of symbols required to represent the integer in Roman numerals, or any other property we wished to define as factors. The GCF approach of this paper to finding similar groups of traces is analogous to this number theory approach to finding similar positive integers, with the relaxation of the arbitrary decision to use prime factors of integers to the more general case.

In addition, it is noteworthy that the GCF approach shown here (1) provides the specific basis for the assertion that a group is and should be identified as a group, and the membership of that group; and (2) can be independently confirmed as to the results regardless of the means by which the GCF analysis is carried out. Thus, regardless of any properties of the algorithm used to generate the results, including any limitations it may have or fault modes present in the implementation or algorithm, results can be tested for truthfulness. For legal purposes, this means that, if properly presented, (e.g., "I used the GCD approach and identified the following similarity groups and their common factors") as long as the results presented are verified as to the presence of the factors in all of the identified members of all of the groups, the statement is strictly true.

2.2 Identifying GCF groups of similar arbitrary factors

Revisiting the potentially faulty assumption that E⇒C, and the arbitrary nature of the use of prime factors identified in the approach to identifying GCF-similar groups of positive integers, we extended this approach to identifying GCF-similar groups of arbitrary factors. This is comprised of a generalization and a specification; (1) the generalization of prime factors and positive integers to arbitrary sequences and traces, and (2) the specification of the E⇒C assumptions

used for analysis.

The generalization from prime numbers to arbitrary sequences; and positive integers to arbitrary traces; is quite straight forward. Other than the method of factoring used in identifying prime factors for GCF-similar groups of positive integers, the approach shown above, in no way relies on properties of numbers. We can simply replace the prime factors and integers, respectively, with defined sequences and traces of our choice, and assuming we provide a method for identifying the sets of sequences present in any given trace as a set, the approach for generation of GCF-similar groups is the same.

Specifying initial E⇒C assumptions is typically done by the use of human knowledge of the behaviors of known or assumed computational mechanisms. For example, and without limit, we might assume that the mechanism that receives electronic mail (email) messages at a mail transfer agent (MTA) faithfully records the Internet Protocol (IP) address contained within the datagrams used to send that message via the simple mail transfer protocol (SMTP) in a "Received:" header that it places at the beginning of the resulting traces it produces. Based on this assumption, we may then develop a parser that identifies sequences consistent with such headers in traces asserted to be the result of email messages sent through MTAs, and call the resulting IP addresses "factors" for the purposes of analysis. Each trace can then be identified with a set of factors through parsing in whatever form, the sets of factors may be grouped so as to form greatest common factor sets, and the groups of traces with those common factors identified as GCF-similar.

2.3 An Algorithm for Finding All GCF-Similar Groups of Arbitrary Factors and its Complexity

As a starting point, just as we must factor all positive integers involved in order to find GCF-similar groups for positive integers, we must somehow parse traces to generate factors used in finding GCF-similar groups of arbitrary factors. As a start to understanding the complexity of parsing, searching for regular expressions is known to be O(n) time, [20] and look ahead left right (LALR) parsing is equivalent complexity (although the construction of an LALR(k) parser is potentially far more complex, once the parser is completed, execution is in linear time). Far more complex challenges meet those working in DFE, including without limit, decryption, steganography extraction, parsing of images and other complex forms not codified as regular expressions or similar syntactic elements, and so forth.

We assume that a parser that has identified and associated all factors with traces (an operation that may take different times depending on the specific parsing requirements associated with the defined factors and traces), and assuming that each trace and associated factor set and factor and associated trace set is accessible in time O(1) and space O(n) where n is the length of the set of parsed tokens plus the length of the list of trace identifiers and traces, by using hash

functions. This is equivalent to assuming that all prime factors have been determined for all positive integers involved in the corresponding number theory problem. The algorithm then follows:

1. For each combination C of more than one trace, find all factors present in all traces in C by taking the intersection of the sets of factors present to produce a map of the GCF of each set of factors to the n-tuples of traces for which it is the GCF.
2. For each set of traces mapped from each set of GCFs identified in step 1, remove sets of traces that are subsets of other sets of traces.

Unfortunately, the size of C that have to be explored to complete step 1 for a set of traces T is:

$$\sum(|T|!k) \text{ for } k \text{ from } |T|\text{-}1 \text{ to } 0$$

which means that completing step 1 using this method for a substantial number of traces is impractical.

The problem of finding GCF-similar groups may also be couched in terms of the covering problem similar to those commonly found in the design of digital circuits. In this formulation, each factor is identified as present or absent (i.e., a different binary digit is used for each of the factors identified through parsing), and the result is a set of bit arrays, one per trace. The GCF-similar group problem is then described as the set of maximum subsets of the bit array (columns) that cover more than one set of traces (rows). This can be thought of as an optimal covering problem in which a set of equations is being sought to cover all of the bits with the minimum set of circuits, but with the added constraint that only maximum prime factors are sought. There are many algorithms that apply to those sorts of problems, but none were identified for finding all GCF-similar groups.

While some other algorithm for accomplishing this may be found, it will not be found in this paper.

2.4 An Approximate Algorithm for More Efficiently Finding GCF-similar Groups of Arbitrary Factors

The approach of this paper is to identify an algorithm that will find a subset of the GCF-similar groups of arbitrary factors in far less time and space than the exact algorithm. For the purpose of parsing and analysis, we will assume that there are a set of characteristics associated with traces and a set of features of each of those characteristics, such that the number of different characteristics is fixed by the parsing approach, but the number of features of those characteristics is limited only by the size and quantity of the traces. For example, a characteristic may be a field within a database, while the features may include, without limit, all of the possible field values, the field type, the ordering of that field within the set of fields, and so forth. Both the presence/absence of the characteristic and the particular features associated with that characteristic are considered "factors" in

our analysis, but the characteristics and the manner in which their values are stored dictate the complexity of the parsing process, while the features dictate the storage required and cause the potential for a large number of different factors. These will be represented as {key, value} pairs, with characteristics treated as pairs with values from {true, false} and features treated as pairs with the characteristic as the key, and the result of parsing as the value.

The approximate algorithm is defined as follows:

1. For each trace ($t \in T$), parse t to create a mapping of factors and traces with each other. ($\forall t \in T\ \forall f \in F$, $txf \rightarrow M$) The mapping is stored in the form of hash tables consisting of: (A) pairs of trace identifiers (TIDs) as keys and pointers to lists of parsed features or elements of {true, false} as values; and (B) pairs of parsed features as keys and pointers to lists of TIDs as values. Thus, as each trace is processed, (1) the lists associated with previously identified factors is augmented to include the new trace and new factors are created as needed; and (2) pointers to the factors associated with the new trace are formed into a list that is associated with the new trace. In the process, the sizes of the value lists for each trace and factor are retained and incremented as appropriate to reflect the total size of the list associated with each hash entry as of this step in the process. There is also a time space tradeoff for storing all pairs of {trace, factor} for later checking.
2. Produce a list of "matched groups" for each of the factors in the content-value list with more than one trace identified, by identifying all of the factors that are common to all of the traces identified with each of those content-value list entries. This is done as follows. For each entry in the factor content-value list with a count of traces in excess of 1, (a) identify the list of traces associated with that entry; (b) retrieve the list of all factors of the first trace identified for that entry from the trace-keyed hashtable, forming a set of those values, and; (c) for each other trace in that entry, remove all values in that set not present in that trace, leaving the set of all common factors of that set of traces, and place the result in a hash table associated with that set of traces (i.e., the "matched groups" hash table). This is done for each entry in the content-value list for which it has not already been done (i.e., for which there is no prior entry in the matched groups hash table). Also store the count of factors and traces contained in each result. Time is also saved if sets of factors for which all of the traces and no others appear are removed from future analysis.

The result is a set "matched groups" composed of pairs of ({factors},{traces}) where, for each pair, each of the set of factors are present in each of the set of traces, and no other trace or factor can be added while retaining this property.

2.5 The Complexity of the Approximate Algorithm

Step 1 is O(|T||F|) for an LALR parser. The creation of the hash tables for step 1 has space and time of O(|T||F|) as well, since constant time is required for placement of {key, value} pairs into a hash table, and the maximum size of each entry is also |F| for the entries with t∈T as keys and |T| for entries with f∈F as keys. For a retrieval time of 1.4 lookups, the size of the hash table is twice the number of entries used, or O(|T||F|).[10] Counting things is O(n) time in the number of things counted and log(n) space in the storage of those counts. Thus step 1 overall is O(|T||F|) in space and time.

Step 2 is requires stepping through the factors, which is O(|F|), and for each associated trace, doing an intersection of the factors associated with that trace. Assuming that all traces are present for each factor, this takes O(|T|) for each factor, with the intersection operation being O(|F|) in space and time. Thus the total maximum time is O(|F|2|T|).

3. RESULTING GROUPS AND LIMITATIONS

The "matched groups" set (M) is a subset of the set of all GCF-similar groups of arbitrary factors. In particular, since each ({factors},{traces}) pair constitutes a set of common factors for a set of traces, and no factor or trace can be added while keeping that property, the matched groups set elements identify sets of traces (groups) for which no factors can be added without causing some trace to have to be removed (greatest common factor). However, it is not necessarily the set of all GCF-similar groups of arbitrary factors.

In particular, because the algorithm identifies all traces (Tx) for a given factor and then identifies the common factors of all of those traces, it is possible that a subset (Ty⊂Tx) of those traces have more common factors than GCF(Tx) that are not included in the overall result ((Ty,GCF(Ty))∉M). However, for (Ty,GCF(Ty)) to not be present in M, it must also be true that for each of the other factors in GCF(Ty), they also appear in another ((Tz,GCF(Tz))∈M) that has this same property. Otherwise, when examining that factor, the group (Ty,GCF(Ty)) would have appeared in M. To date we have not identified an algorithm that would allow the detection of such cases that is more efficient than the original problem of finding all GCF-similar groups, that being ∑(|T|!k) for k from |T|-1 to 0

3.1 Summary of the resulting groups

The result of this algorithm is a set of groups of characteristics and features (i.e., factors) associated with sets of traces, in which each group has the largest collection of traces for which all of those factors present, and the set of all traces for which at least one such factor is present. Or, as the title of the paper indicates, the "greatest common factors" for each group consisting of more than one trace. The resulting set also produces the largest sets of traces for each of the factors present in more than one trace, so that for the identified set of factors, no larger set of traces contains all of those factors.

3.2 The challenge of identifying factors

It is noteworthy that there are limitations of this approach, in particular, because of the nature of the way in which factors are defined. For example, this approach would be extremely inefficient for searching for all values of an integer value in a range where that range is not specified in advance of the analysis, or for other sorts of operations where the number of factors grows large. This is because the space and time grow as the number of traces times the number of factors. The total number of factors potentially identifiable with a trace is O(b!) where b is the number of bits in the trace [3] and as the number of traces groups, this also grows factorially. Notionally, the way to identify factors for analysis is to identify characteristic behaviors of known causes. For example, the mechanisms that produce traces associated with specific email clients can be identified and differentiated and used as the basis for defining factors that will differentiate between a set of know mechanisms. This the notion of similarity will be relative to the different sorts of known mechanisms and dissimilarity will imply that different known causes may have produced the dissimilarity and refute any claims that dissimilar known mechanisms caused them. Thus by choosing factors to differentiate known causes, the mechanisms M: C⇒mE is partitioned, leading to more certainty surrounding the E⇒C assumption.

3.3 Limits on use as an exploratory tool

This method performs a single sequence of calculations with a potentially significant runtime and produces a set of GCF-similar groups of arbitrary factors. While the result can be readily searched for interactive analysis, in a case where the runtimes are significant and an exploration of potential factors is desired, the approach will take significantly longer than an incremental approach in which a new factor is considered and all consistent traces are searched and identified. An incremental process may be used to update Step 1, and there may be an incremental approach to updating step 2, but none is currently known.

A simple exploratory approach based only on the use of a database does not automatically identify combinations of factors and sets of traces with these common factors, and is thus significantly limited. A combined approach, in which a database is used for exploration and GCF analysis is used for creating groups to be considered can be a useful combination for exploratory analysis efforts.

3.4 Applicability for unstructured data

The greatest common factor analysis method is typically applicable for structured or semi-structured data, or on the results of an algorithm applied to unstructured data to generate relevant factors. For unstructured data, a significant amount of time may be required to parse features, such as the presence or absence of a "table", "skin tones", areas of certain sizes, etc. Human examination is often far more efficient for small collections, and automation for such unstructured data is quite limited today. However; anything that can be characterized as features and

parsed can be factored and GCF-similar grouped through this approach.

3.5 Applicability relative to n-tuple analysis and proximity

Among the more widely applied methods for similarity analysis are the n-tuple and proximity approaches identified above for searching. The GCF-similar grouping approach is not directly reconcilable with the n-tuple approach because of the large number of factors associated with possible groupings and sets of n-tuples. To get a sense of this, if there is an ordered 5-tuple within a trace, that also means that there are either 2 or 5 ordered 4-tuples ({(1,2,3,4), (2,3,4,5)} or {(2,3,4,5), ((1,3,4,5), (1,2,4,5), (1,2,3,5), (1,2,3,4)} depending on proximity limits), up to 10 2-tuples {(1,2), (1,3), (1,4), (1,5), (2,3), (2,4), (2,5), (3,4), (3,5), (4,5)}, and 10 3-tuples {(1,2,3), (1,2,4), (1,2,5), (1,3,4), (1,3,5), (1,4,5), (2,3,4), (2,3,5), (2,4,5), (3,4,5)}. All of these may apply to different sets of traces, and thus all must be included in F in order to get an accurate GCF result for each of the different n-tuple lengths. If n-tuples and proximity features are calculated as part of the parsing process and fed into the GCF-similar grouping approach for inclusion in the grouping results, performance problems will likely result, since time and space go as |F|2.

A more effective approach has been identified as the completion of the GCF-similar grouping, followed by the inclusion of n-tuple and proximity analysis results within each of the applicable GCF-similar groups. In this approach, the n-tuple or proximity analysis is done on each of the groups, and this means that these analyses are limited to those groups rather than the entire set of traces.

While this does not produce the full set of similarities with respect to those approaches across the entire set of traces, it does produce these results for each of the identified groups. Thus additional similarities can be identified with the GCF-similar groups, and this both increases the number of similarity factors, and does so without altering the GCF nature of those groups. The reason the groups are still GCF-similar groups is that the addition of more common factors to such a group cannot increase the groups size, which was already maximal, and cannot reduce its size, because all of the added factors are true for all traces within the group.

3.6 A sample result

Results of analysis, as provided by an implementation in LISP, are demonstrated by the following snippet from [16]:

> 63 4 "User-Agent" "User-Agent: Mozilla/5.0 (Macintosh; U; PPC Mac OS X Mach-O; en-US; rv:1.6b) [REDACTED]") ("LOCContent-Type" . "14") ("From: [REDACTED]@[REDACTED]") ("LOCFrom" . "6") ("LOCIn-Reply-To" . "13") ("LOCLines" . "17") ("MIME-Version" . "MIME-Version: 1.0") ("LOCMIME-Version" . "9") ("LOCMessage-ID" . "16") ("LOCNNTP-Posting-Date" . "4") ("NNTP-Posting-Host" . "NNTP-Posting-Host: [REDACTED]") ("LOCNNTP-Posting-Host" . "18") ("Newsgroups" . "Newsgroups: [...]") ("LOCNewsgroups" . "10") ("LOCPath" . "3")

("LOCReferences" . "12") ("Subject" . "Subject: Re: [REDACTED]") ("LOCSubject" . "11") ("LOCUser-Agent" . "7") [...] ("Received-IP" . [REDACTED]) ("Traces:" 250607 250565 250506 250489)

Much of this result is redacted for space and privacy purposes, but these 4 postings were similar in 63 factors.

4. SIMILARITY METRICS

In attempting to provide metrics for similarity, various authors have created measures of different sorts, typically for pairs of items being matched one against the other, or for matching of a regular expression or similar description against a set of items, such as for search engines. For example, two items that are identical in every way could reasonably be called 100% similar, and if a string "12374382302398" is found within one of the items being searched, the item could be reasonably reported as 100% similar to the criterion that it contains that sequence. However, for less trivial situations, similarity metrics may be problematic, and similarity metrics for groups have not, as far as we have been able to identify, been developed.

Using the GCF method described herein, some obvious metrics appear. Statements such as "Similar in n factors", or "Have x common factors", may be applied to the groups created. For example, to e-mails may be identified as having 18 common factors in their headers. But while this method may turn normal metrics (the factors) into interval metrics, our ability to count does not meaningfully address the question of counting against what.

While there may be a temptation to make statements such as this group is similar in 18/360 factors, the notion that they are 5% similar is essentially meaningless in context. In fact, it is relatively easy to create factors such that we can drive those percentages up or down for any given group. For example, we can remove factors that are not part of the results we have determined and produce "100%" similarity results, once we know what to choose as factors.

Furthermore, the question of how to weight factors and/or groups of factors depends on the particulars of the application. For example, if we are trying to attribute telephone calls to callers, and we have groups that had identical calling phone numbers, this would seem in most cases, to be more important than the fact that we may have large groups of calls of particular durations. Thus, for the attribution of calls to callers, phone numbers may be weighted more highly than call duration. However, if we are seeking text messaging, then call duration of relatively short time scales will almost certainly be a far better indicator than the phone number from which the call was made.

As a result, it is likely the best approach at this time to define metrics such that the features of interest to the application are weighted prior to analysis, ratio metrics are eschewed in favor of intervals, and to the extent that ordinal metrics may be applied to factors, a POset that may be developed to rank results with identical

interval values. Without further information about specifics of the situation, the use of approaches like weighted sums may be more misleading than helpful.

5. A PRACTICAL EXAMPLE

The initial implementation of this approach was applied to identifying groups of messages in message archives. The implementation was tested over time with different message archives, and eventually applied in a legal matter. [16]

5.1. The matter at hand

In the matter at hand, Plaintiff asserted, in essence, that Defendants slandered Plaintiff by, without limit, making statements about postings to newsgroups and unauthorized accesses to attorney-client privileged electronic mails (emails). Because the truth is an absolute defense against claims of slander, Defendant's expert sought to apply similarity analysis along with other methods to demonstrate that particular postings were attributable to third parties at issue. Because the newsgroups at issue contained more than 200,000 postings manual analysis was identified as infeasible, and automated methods were applied.

5.2 The GCF-similarity group approaches and results

In particular, and without limit, these methods included the analysis of the newsgroup postings to identify GCF-similar groups, and the subsequent manual review of the results to identify how the groupings produced related to the issues in the matter.

Parties asserted that a particular set of postings to two newsgroups were of import. These postings were analyzed to identify GCF-similar groups and found to demonstrate particular groupings associated with "posting account", IP addresses, software configuration of posting computers as potentially depicted by "user-agent" or similar fields, and the sequencing of headers within the header area.

The entirety of the available postings on the applicable newsgroups were downloaded from an Internet Service Provider (ISP) who was independent of the parties in the case and who regularly gathered and provided these records as part of their services for their clients, and depended upon them for normal business operations. Thus these records are, presumably, admissible as evidence under the normal business records exception to the hearsay prohibition on evidence.

The newsgroup messages were parsed based on the same criteria as the asserted postings and analyzed in the same manner for GCF-similar groups. The resulting groups were compared to similar groups identified for the asserted postings and then examined by an expert to help attribute the asserted messages to sources. In this case, the placement and ordering of headers and n-tuple analysis that can be added to augment analysis were not applied because the increased computational complexity drove the space and time beyond available resources. Rather, these factors were added after the GCF-similar groups were identified and particular groups found to be probative to the matter at hand.

In the particular matter at hand, the headers with traces indicative of, without limit, "posting account", IP addresses, characteristics of the software configuration of the computer making the posting as depicted by "user-agent" or similar fields, and the sequencing of headers within the header area from the entirety of the large newsgroups were compared to those of the asserted messages, and groupings found within the newsgroups as a whole that showed similarity in the sense of common factors in these areas.

The similarity analysis showed that there was one posting that was self-asserted to be from one of the parties to the case and that that party did not deny posting, and that was identical in the relevant areas to asserted postings. Out of about 200,000 total postings, only 64 contained the particular sets of factors, all but one of those were identified as the asserted postings prior to the analysis, and the remaining posting was the one self-identifying as the party to the matter. On a different forum, an analysis was done that also identified factors of similar sorts, and in that forum, postings self-asserting as attributed to the same party also had many of the same sets of factors in the GCF-similar groups associated with the same party, providing an independent path for attribution.

The IP addresses of the asserted postings also partitioned those postings into GCF-similar groups, and those groups were compared to GCF-similar groups in the newsgroups as a whole and GCF-similar groups from the other forum. Again, these groups were consistent with the IP addresses of the asserted party and were also consistent with the IP addresses used by the spouse of that party, providing yet another confirmation of attribution.

5.3 Addressing the E⇒C assumption fallacy

As discussed earlier, the E⇒C assumption is a fallacy in that there are clearly cases when different causes produce similar effects. As a result, a GCF-similar group does not necessarily imply the same causes produced the same effects reflected in the set of factors in the traces. In addition, the lack of all factors being identical could be considered a refutation if this was inconsistent with the hypothesis about common cause.

In this particular case, some of the IP addresses identified were also determined, through warrants and credit card information, to have been assigned to and paid for over a period of years, by the spouse of the previously identified individual; the user account name, address, and other contact number of the account used to make the asserted postings were also those of the same party; and other information subsequently discovered confirmed these results.

5.4 Presentation of the results

Thus the CGF-similar groups of posting and subsequent investigation showed a far stronger attribution of postings to their source and demonstrated results consistent with the asserted causal chain. Ordinal metrics were applied in context (e.g., 64 out of 200,000), and statements were made using the applicable forms.

(e.g., "Traces containing the identified posting accounts were found to be similar in 37 factors, contained identical header sequences, and were found present in only 64 out of 200,000 messages, one of which appears to be from [party] and the rest forgeries identified with the issues in this case")

While the overall matter had other more complex elements involved in the attribution process, the similarity results associated with the GCF approach and the metric of count of similar postings within the overall corpus under consideration provides a powerful argument in favor of a particular interpretation, and in this case, that interpretation is consistent with an attribution at issue in the case.

6. ADDITIONAL MATERIAL AS TO THE PRACATICALITY OF USE IN THE CASES CITED

Per editorial comments on the original submission of this paper, additional detailing of the cases identified above were requested. This particular case[10] involved downloading of usenet group postings from over the entire period since the start of the groups till the time at which the case was initiated. Since there is no standard method for doing forensically sound downloads, the postings were downloaded using the methods available from the Internet Service Providers (ISPs) who, at that time, permitted usenet group downloads from servers supported by each participating ISP.

Ignoring the particulars of the download process, the protocols used provide the headers and bodies of the postings. In this case, the headers formed the structured content which could then be examined and analyzed for similarity using the present methods. In particular, usenet news posting headers included, without limit, the following headers in the formats associated with the applicable requests for comments (RFCs) to usenet news group postings:

Path, From, Newsgroup, Subject, Date, Organization, Lines, Message-ID, Mime-Version, Organization, User-Agent, X-Complaints-To:, and X-HTTP-UserAgent"

The analytical process to generate greatest common factor sets was run on the approximately 200,000 downloaded messages to identify the set of all such greatest common factor (GCF) sets. This process took approximately 8 hours of real time to complete, including parsing out of headers and header components based on investigator-specified criteria (which in this case involved taking account of each header, its location, content, relevant IP addresses, and several other similar header characteristics).

In this particular case, a set of identified messages were specified as relating to the potential perpetrator, and as a result those GCF sets containing elements of this set of previously identified messages were sought from the resulting output using the "grep" command, which owing to the fact that the identified messages had unique message identifiers and that the output of the particular GCF set application used a single line to describe each GCF set it found, allowed the pre-

existing "grep" program to be applied for that purpose. Furthermore, since this takes only linear time with the number of GCF sets, it is time efficient and does not cause an overall increase in the computational complexity of the method in use.

The output of the existing program provides the set of all messages (by providing a list of filenames associated with each of those messages) for which the identified GCF set (identified by a list of the set of common factors they all have) are present, and indicates additional information such as set sizes. In this particular case, out of the several hundred thousand postings, the set of identified postings were revealed as the largest GCF set involving any of the identified postings, and was thus directly usable in further examination. In order to proceed with subsequent examination, the identified 64 postings were examined using the "diff" command, each identified factor was sought and found in each of the set elements and sought and not found as a group in other postings. This independent verification of results was done.

In this particular case, in addition to the many similarities that might reasonably be interpreted as coming from a wide range of systems, and the fact that many of those factors listed factors were present in other larger sets of postings (such as the locations of some of the more common headers), the factors that made this particular set more clearly distinguishable based on their lack of presence in other postings were fields typically related to specific configuration details of specific systems. For example, the fields indicating browser header information contained identical operating system, library, and other similar factors that were not present in any other postings and were present in all of the postings in the GCF set.

Thus the opinion of the examiner in this case was that these particular factors were consistent with the same system being used in a similar time frame to post the many postings identified as at issue in the matter at hand and the one such posting that was self-identified as from one of the parties to the case, and which that party refused to disassociate himself with or deny he had posted.

From a practical standpoint, identification of such a correlation between 64 out of 200,000 or so files by manual means is likely infeasible, and fraught with the potential for errors. Attempts to "eyeball" it and identify things to look for would potentially be useful, but this would then require that, once such factors were found, additional manual efforts or programming would be required to determine the base rates, and those methods would only be as efficient as the programmer and method used to look for them. The present method automatically generates GCF sets that are unique and maximal, thus producing 0 base-rate conditions for the presence of all GCF factors together elsewhere in the corpus.

As a practical matter, running such analysis on messages or similar structured, artificially structured, or partially structured content is relatively simple to do and can be applied as a standard process which similarity analysis is a feasible method for identifying things to examine in more detail or when attribution is at issue. To

the extent that it is revealing with regard to similarity or helpful in attribution, the examiner can readily apply the results based on their knowledge, experience, expertise, skills, and education to make judgments about the materiality of the results.

In this particular case, the GCF analytical process helped to produce such results as:

> Postings containing the "X-HTTP-UserAgent: Mozilla/4.0 (compatible; MSIE 7.0; AOL 9.0; Windows NT 6.0; SLCC1; .NET CLR 2.0.50727; Media Center PC 5.0; .NET CLR 3.0.04506; InfoPath.2),gzip(gfe),gzip(gfe)" header are found 64 times in identified messages, this header is consistent with a computer using a Mozilla version 4.0 Web browser on a computer with a Windows NT 6.0 operating system, using AOL 9.0, Microsoft Internet Explorer version 7.0, and with a variety of other quite specific versions of different related software packages present, it was present in only 64 of more than 200,000 such postings examined in this case, all also have 12 other commonalities in headers, 36 have IP addresses used in other postings by the identified party to the matter, all 64 postings are identified as part of the behavior at issue in the case, each of the 4 different "From" addresses identified as possible intentional forgeries have "From:" addresses with IP addresses used at contemporaneous times by the identified party to the case in postings to other sited under his own identity, and the substring "Mozilla/4.0 (compatible; MSIE 7.0; AOL 9.0; Windows NT 6.0; SLCC1; .NET CLR 2.0.50727; Media Center PC 5.0; .NET CLR 3.0.04506" was also present in the records of postings to another site by the same party in the same time frame while no other authorship was associated with any postings containing the same strings.

Both for identifying potential sources of similarity and for identifying other commonalities once a particular factor of interest was identified, the GCF-similar group approach produced a dramatic reduction in time and effort and a far greater capacity to rapidly check out working hypotheses and identify specific sets of relevant traces than manual or perviously applied automated means.

7. SUMMARY, CONCLUSIONS, AND FURTHER WORK

While it is clear that the GCF-similar group approach is still in its infancy and that there has been relatively little published work in the creation of such similarity groups as opposed to testing for similarity of a specification against a set of items or one item against other members of the set, it is also clear that a thorough examination of all similarity measures is infeasible.

As a result, a computational compromise is required in order to find ways to identify similarities and provide metrics for those similarities for different purposes. The computational compromise of this paper consists of identifying an O(|T||F|2) algorithm that fails to identify a specific class of GCF-similar groups of

identified factors while producing the remaining groups.

The basis for the use of factors in similarity analysis under the GCF method includes a fundamental assumption that the similarities produced in traces are or may be the result of the use of the same or nearly identical mechanisms used to produce those traces, the E⇒C assumption, which is known, in general, to be a fallacy. This assumption is easily violated in conditions where the traces at issue are not under the control and in the possession of an independent third party that keeps them with integrity, or in cases where intentional efforts are made by knowledgeable and skilled parties to forge traces. However, it is also noteworthy that it is not as easy as it may seem to create perfect forgeries, particularly in environments in which only a limited portion of the available traces can be influenced by the party. Further, the intentional selection of factors that differentiate causal mechanisms is useful in eliminating alternative explanations.

This method also has the fundamental problem, as do all such methods, of requiring an initial parsing along with all of the assumptions that go with that parsing. Again, this traces back to the E⇒C assumption and the notion that mechanisms produce predictable results. For this reason, it appears to be most suitable today for analysis of structured headers and meta-data.

The present algorithm is likely not the most efficient one that can be developed in this arena, but rather, it is the first algorithm that has been developed for this purpose. The concept of greatest common factor appears to be worthy of further pursuit, and in addition to seeking a better implementation, the concept itself, in terms of the manner in which factors are developed and applied, and in terms of the creation of better or more meaningful metrics, also appears to be worth further research.

Research into the effectiveness of this approach for unstructured data would be meaningful, in that while we know that the method will work, we don't know how useful it may be in practice. The challenge of identifying factors and the implications for parsing complexity remains, as does the issue of improved integration with n-tuple, proximity of terms, and other similar approaches. Research into low-complexity enhancements to coverage of the full GCF-similar group set may prove fruitful, and the production of more efficient algorithms that reduce the |F|2 term, would allow the extension of this method to far larger factor sets than can be used today, but there may be lower bounds that limit the ultimate effectiveness of this approach.

REFERENCES

[1] E. Locard, "The Analysis of Dust Traces", *The American Journal of Police Science*, (1930), V1:276-298. [This is one of the most referenced papers in forensics, and is considered the defining paper in this area by many subsequent authors.]

[2] F. Cohen, "Digital Forensic Evidence Examination", 2009, ASP Press. [This book introduces a physics of digital systems and explores issues in examination of digital forensic evidence for legal purposes, including analysis, interpretation, attribution, reconstruction, and presentation. It also includes a substantial review of the literature in related areas.]

[3] F. Cohen, "Analysis of redundant traces for consistency", *IEEE International Workshop on Computer Forensics in Software Engineering* (CFSE 09), Seattle, Washington, USA, July 20-24, 2009. [This paper provides complexity results for analysis of traces and extends the use of redundancy for trace analysis for the identification of consistency and inconsistency of traces and events.]

[4] T. Stallard and K. Levitt, "Automated Analysis for Digital Forensic Science: Semantic Integrity Checking", *ACSAC*-2003. [This paper identifies the use of redundancy for testing hypotheses with regard to digital forensic evidence.]

[5] F. Cohen, "Two models of digital forensics examination", IEEE/SADFE-2009, *Fourth International IEEE Workshop on Systematic Approaches to Digital Forensic Engineering*, Oakland Conference, Oakland, CA, USA, May 21, 2009. [This paper identifies a model for performing analysis of issues related to the examination of digital forensic evidence.]

[6] K. Popper, The Logic of Scientific Discovery (1959), Hutchins and Company, London. ISBN10: 0415278449.. [This book is considered a defining standard for philosophy of science in terms of identifying the principles and properties of confirmation and refutation in the scientific arena.

[7] Federal Rules of Evidence Rules 701-706. [These are the rules used by the US Federal and many state court systems to determine admissibility of evidence and expert witness credentials and testimony.]

[8] Daubert v. Merrell Dow Pharmaceuticals, Inc. 509 US 579, 125 L. Ed. 2d 469, 113 S. Ct. 2786 (1993). [This is one of the key cases that defines the standard for admissibility of evidence based on Supreme Court precedent.]

[9] Committee on Identifying the Needs of the Forensic Sciences Community, "Strengthening Forensic Science in the United States: A Path Forward", ISBN: 978-0-309-13130-8, 254 pages, (2009).;

Committee on Applied and Theoretical Statistics, National Research Council. [This report outlines the current limits of scientific evidence in the US legal system and identifies a need for increased scientific rigor in many aareas.]

[10] D. Knuth, "The Art of Computer Programming, Volume 3: Sorting and Searching", ISBN 0-201-03803-X, Addison Wesley, 1973. [This book summarizes a wide array of research in computer science, and is the third in a series of books by this author that are highly regarded in clarifying the nature of results in computer science.]

[11] S. Brin and L. Page, "The anatomy of a large-scale hypertextual Web search engine", *Computer Networks and ISDN Systems*, Volume 30, Issues 1-7, April 1998, Pages 107-117. (see also an extended version at http://infolab.stanford.edu/~backrub/google.html) [This paper defines the architecture used to provide for the initial Google search engine and identifies many of the properties and features of many search engines since that time.]

[12] C. Chaski, "Who's At The Keyboard? Authorship Attribution in DigiEvidence Investigations", International Journal of Digital Evidence, V4#1, 2005. [This paper summarizes results in attribution relating individuals to actions based on behavioral characteristics and identifies the extent to which the US legal system to date has accepted such results in court cases.]

[13] T. Pedersen, "Computational Approaches to Measuring the Similarity of Short Contexts: A Review of Applications and Methods", Journal of Intelligent Systems (Special Issue : Recent Advances in Knowledge-Based Systems and Their Applications), 17(1-3), 37-50, 2008. http://www.d.umn.edu/~tpederse/ [This paper summarizes results in examining n-tuples and proximity measurements for natural language processing to determine similarity and attribute word sequences to authorship.]

[14] M. Corney, "Analysing E-mail Text Authorship for Forensic Purposes", Masters Thesis, Queensland University of Technology, March, 2003 [This thesis examines using a variety of classifiers with output fed into a Support Vector Machine (SVM). The approach is to compare a specific email to an SVM model built from a corpus of emails with known provenance e.g. given 20 emails from each of A, B and C, compare a

new email to those models to see which author it is appears to belong to.]

[15] Bomze, M. Budinich, P. Pardalos, and M. Pelillo, "The Maximum Clique Problem", *The Handbook of Combinatorial Optimization* (http://reference.kfupm.edu.sa/content/m/a/the_maximum_clique_problem__265525.pdf). [This book chapter summarizes mathematical results related to the identification of cliques, a problem loosely related to the GCF group problem addressed by this paper.]

[16] Susan Polgar vs. United States of America Chess Federation et. al. Case # 5-08CV0169-C. [This is a current legal matter in which the results of analysis have been publicly released. It was still pending at the time of the original submission of this paper, is related to several other pending cases, including a Federal felony case against a related party but this particular matter has now been settled. It applies the analytical techniques identified in this paper in support of attribution.]

[17] J. Hunt and M. McIlroy, "An Algorithm for Differential File Comparison". Computing Science Technical Report, Bell Laboratories 41. (see http://www.cs.dartmouth.edu/~doug/diff.ps) June 1976. [This paper defined the algorithms used in the Unix "diff" program and introduces the problems associated with identifying maximum matching sequences.]

[18] D. Maier (1978). "The Complexity of Some Problems on Subsequences and Supersequences". J. ACM (ACM Press) 25 (2): 322–336. [This paper summarizes results in matching and related problems for pairs of sequence, including complexity results associated with a wide range of related problems and details of algorithms and their limitations.]

[19] L. Bergroth and H. Hakonen and T. Raita (2000). "A Survey of Longest Common Subsequence Algorithms". SPIRE (IEEE Computer Society) 00: 39–48. [This paper summarizes results in the particular specialty of identifying longest common sequences, and is the evolution of the work on identifying differences (and similarities) related to the "diff" and similar sorts of programs.]

[20] P. Weiner, "Linear pattern matching algorithm". 14th Annual IEEE Symposium on Switching and Automata Theory: 1-11. (1973). [This paper demonstrates an approach to doing linear time matching of regular expressions to strings based on the creation of a finite state machine that optimally implements a matching device. The device can then be

implemented in software with linear time detection of these expressions. The same method can be applied to LALR parsing and a wide range of other similar problems that are equivalent in relevant ways to regular expression parsing.]