Unsupervised clustering of file dialects according to monotonic decompositions of mixtures

Michael Robinson
Department of Mathematics and Statistics
American University
Washington, DC
Email: michaelr@american.edu

Tate Altman
Department of Mathematics and Statistics
American University
Washington, DC
Email: ta8427a@american.edu

Denley Lam
BAE Systems FAST Labs
Arlington, VA
Email: denley.lam@baesystems.com

Letitia W. Li
BAE Systems FAST Labs
Arlington, VA
Email: letitia.li@baesystems.com

***Abstract*—This paper proposes an unsupervised classification method that partitions a set of files into non-overlapping dialects based upon their behaviors, determined by messages produced by a collection of programs that consume them. The pattern of messages can be used as the signature of a particular kind of behavior, with the understanding that some messages are likely to co-occur, while others are not.**

We propose a novel definition for a file format dialect, based upon these behavioral signatures. A dialect defines a subset of the possible messages, called the required messages. Once files are conditioned upon a dialect and its required messages, the remaining messages are statistically independent.

With this definition in hand, we present a greedy algorithm that deduces candidate dialects from a dataset consisting of a matrix of file-message data, demonstrate its performance on several file formats, and prove conditions under which it is optimal. We show that an analyst needs to consider fewer dialects than distinct message patterns, which reduces their cognitive load when studying a complex format.

I. Introduction

While considerable effort has been expended to formalize what compliance with a format specification means, the behavior of programs when files are consumed is what defines the end-user experience of a given format. A behavioral understanding of file formats has the advantage that it is amenable to a statistical perspective, wherein one can ascribe the conditional probability that a particular file will elicit a particular behavior given that other behaviors have already been observed.

Because behaviors are characterized by a large number of possible features, it is important to organize them into models that can be more easily understood by an analyst. To organize this information most effectively, what a file format analyst needs is not necessarily captured by the most accurate model of these behaviors, but rather one that is both accurate *and* easily explainable. We argue that explainable models are parsimonious. They contain the minimum number of dialects necessary to characterize the observed behavior.

The behavioral perspective aligns neatly with the discipline of test-driven design, since files that elicit unwanted behaviors can easily be identified as test cases. As such, curation of format-compliant file datasets is an important task for a file format analyst. Files that are supposed to comply with a given *ad hoc* format specification may in fact fall into one of several *dialects*, in which different patterns of behavior can be observed. Managing the behavioral differences between dialects is a source of trouble when one is attempting to construct programs to consume files of a given format. To obtain adequate test coverage, one must ensure that all dialects are present in the test samples, which means that an analyst must first know which files in their dataset comply with which dialects. By partitioning the file format into dialects, parser developers following the LangSec approach can develop grammars covering each dialect to develop more comprehensive parsers, or they can explicitly choose which dialects of a format their grammar should cover.

A. Contributions

This paper proposes an unsupervised classification method that partitions a set of files into non-overlapping dialects based upon their behaviors, which are measured by the occurrence of a collection of Boolean features, called *messages*. The pattern of messages can be used as the signature of a particular kind of behavior, with the understanding that some messages are likely to co-occur, while others are not.

Our method is based upon a novel statistical definition for a behavioral dialect. Each dialect defines a subset of messages, called the *required messages*, that satisfies the following statistical assumption. Once files are conditioned upon a dialect and its required messages have occurred, the remaining messages are statistically independent. The implications of this definition are detailed in Section II.

Our definition of a dialect leads to a greedy algorithm (Algorithm 1) that deduces candidate dialects from a dataset

consisting of a matrix of file-message data. The decomposition of a dataset into all observed dialects is embodied by the constructive proof of Proposition 3 in Section IV.

The method we propose is able to work with files of any format, provided enough messages are available. To highlight this fact, in Section III we explore our method's performance on three vastly different formats: a tabular data (CSV), free-form documents (PDF), and images (NITF).

Finally, we establish (Theorems 1 and 2 in Section IV) that our method is optimal in the sense that it yields the least number of extraneous dialects once certain reasonable statistical assumptions are satisfied. The proofs of these theorems appear in the Appendix, Section VI.

B. Limitations

Our methodology relies upon good message coverage for the format under consideration. If messages are not elicited by the behaviors of interest, then files which exhibit these behaviors cannot be detected. Fortunately, most available parsers for the *ad hoc* formats we consider in Section III produce copious output to `stderr` and `stdout`. This output is sufficiently standardized that regular expressions (regexes) can be used to collect the output into messages.

If the set of messages being used is known to a malicious actor, our approach might be subverted by crafting files to avoid producing certain messages. Message regexes can be constructed in a semi-automated way, outlined in detail by [1], so it is not difficult to obtain enough messages of sufficient diversity to prevent files from evading proper classification. Messages can also be made that correspond to system calls, resulting in additional behavioral diversity [2]. Anecdotally, it seems difficult to construct files that avoid *all* messages of a certain type.

To manage the complexity of our model, we relax independence into a monotonicity condition. As a result, instead of obtaining the best mixture decomposition, we merely obtain bounds upon it, see Theorem 2. Nevertheless, this kind of bound is not generally available for the usual statistical tools for inferring mixtures. Because our implementation uses a greedy algorithm, there is some ambiguity that results in our candidate dialects, though this is fairly benign. When several messages are statistically dependent they need not all be chosen as required for a dialect, see Lemma 8 for details. Our algorithm scales linearly in the number of dialects and quadratically in the number of message patterns.

We broadly assume that the dialects do not overlap even though it is well known that files can comply with multiple unrelated formats simultaneously. Nevertheless, it seems that our model supports overlapping dialects in practice. Specifically, our statistical model yields probabilities that a file is of each candidate dialect. These probabilities can be interpreted as allowing a file to be a mixture—in itself—of several dialects.

C. Related work

This paper continues a line of work presented over the past few years in the LangSec community that takes a statistical look at format specifications [1; 3; 4]. In contrast to the hypothesis in [1], that *messages are independent when conditioned upon dialect*, the present paper additionally conditions upon a set of messages that are required for each dialect. When messages from different parsers are combined, some of these messages are effectively identical features. For instance, two parsers may emit the same kind of syntax error, resulting in two separate but statistically dependent messages. This kind of behavior was ignored [1] even though it is a direct violation of the assumption of independence. By conditioning upon one, the other, or both of these messages, we can restore independence.

Statistical format analysis appears to be a minority viewpoint, because traditional file format analysis uses the structure of file *contents* rather than the *responses* of parsers to those contents (for instance, see [5; 6; 7; 8; 9]). Nevertheless, statistical features based upon file actions has also been used to identify certain malicious behaviors [2].

We take inspiration from [10], in which 39 dialects of CSV files were found. Our methodology was applied to a simple random sample of the same dataset, wherein we find a somewhat coarser collection of 14 dialects being most common, though numerous less common ones are also detected (see Section III-A).

Our statistical model is a special case of a non-parametric independent mixture model. Independent mixture models are very well studied in the literature, with many algorithms that have deep theoretical backing (for instance [11; 12] among many others). Unfortunately, these algorithms tend to make assumptions that are inappropriate for the context of file-message data, such as assuming the components are of a known distribution or that the number of mixture components (dialects) is known from the outset.

Expectation maximization is a common tactic to avoid making assumptions about the underlying distributions, at least when the number of mixture components (dialects) is known and/or small. The resulting family of techniques for inferring the components of a mixture using expectation maximization is generally called latent class analysis (LCA). Under broad theoretical conditions, LCA yields a decomposition of the empirical distribution into components. In fact, it appears that the use of LCA for inferring file dialects would be novel. Nevertheless, *we do not use LCA in this paper* to determine dialects. LCA does not provide as much insight as the method we present. Expectation maximization—probabilistic accuracy of the model—is only one objective when exploring a file format. Explainability of the results is also important, and LCA provides no theoretical guarantees about the correspondence between the discovered dialects and the latent ones. Additionally, while LCA may be computationally feasible, it does not scale as favorably as our greedy algorithm [11, 1.3.1].

Our problem involves hundreds of features (messages) with an unknown and potentially large number of classes (dialects). This means that the problem of inferring dialects using LCA is underdetermined. Our methodology is able to turn the underdetermined nature of the problem into an advantage be-

cause we take a unique mathematical perspective, based upon *partially ordered sets* and the *Dowker complex* [13]. Recent work has connected the Dowker complex to tabular data [14] and to formal concept analysis [15]. We show that there is an ordering among the candidate dialect decompositions that are consistent with the data. Moreover, there is a unique coarsest decomposition of the data into dialects which can be found by a greedy algorithm.

Expectation maximization can be applied to the monotonic decompositions described in this paper as well. Our method finds the decomposition that simultaneously has the largest dialects (Theorem 1) and the fewest number of dialects (Theorem 2). Together, these two properties make the resulting decomposition the easiest to explain. Unfortunately, expectation maximization can only find one decomposition with no guarantees about the dialects it finds.

II. Statistical model of file format dialects and their behaviors

Our data consist of a set of files F, which when parsed by a variety of programs may yield any of a certain set of messages M. For each file, each message $m \in M$ either does occur (in which case we say that $m = 1$), or does not occur (expressed as $m = 0$). Although parsing a given file is (usually!) deterministic, we can model the likelihood of a given message occurring as a probability $P(m = 1)$. A probabilistic model avoids handling specific files individually, so we rarely need to handle the set F of files directly.

In [1; 3], it was shown that the joint probability distribution of a set of messages can identify certain files of interest. In this probabilistic setting, an event consists of a *message pattern*, which is a subset $K \subseteq M$ of messages that might occur. We notate the joint probability of obtaining exactly the messages in K as $P(K)$.

Studying arbitrary joint probabilities on their own is fraught, though the data often support useful statistical assumptions that provide theoretical traction. In [1], it was assumed that messages for files within a given subset $A \subseteq F$ were independent. While this is a reasonable assumption when messages are semantically unrelated, it is not appropriate when some of the messages are related to each other. In this article, we take a more refined approach. Specifically, each dialect specifies a set of *required messages* that must occur for a file to be considered part of that dialect. In each dialect, non-required messages may be correlated with each other only in so far as they are correlated with the required messages. Non-required messages in a dialect may be correlated in the joint distribution for the whole data, though they become independent *once conditioned upon the required messages*.

Definition 1. A *dialect* is a subset of files $A \subseteq F$, to which one can ascribe a subset of messages $R_A \subseteq M$, such that once conditioned on both R_A and A, the remaining messages are independent. The subset R_A is called the set of *required messages* for dialect A. We will call the subset of all message patterns that contain R_A the *support of the dialect* A.

If a message pattern K is not in the support of the dialect A, then $P(K|A) = 0$, which asserts that no files in dialect A will exhibit the message pattern K.

Explicitly, the probability of obtaining message pattern $K \subseteq M$ for a file in A is of the form

$$P(K|A) = \begin{cases} 0 \text{ if } R_A \not\subseteq K, \\ P(R_A|A) \prod_{k \in (K \cap R_A^c)} P(\{k\}|A) \times \\ \quad \prod_{k' \in (K^c \cap R_A^c)} (1 - P(\{k'\}|A)) \text{ otherwise.} \end{cases} \tag{1}$$

Not every subset of files will yield a subset of messages that can serve as the required messages for a dialect. Additionally, required messages can be overlapping even if the files are disjoint. Our Algorithm 1 will have trouble discerning between two dialects with *identical* sets of required messages, though this is not likely to happen if message sets are diverse enough.

Conversely, there may be several ways to partition the set of files F into a collection of disjoint dialects. Our methodology exploits this ambiguity by organizing all of the possible ways to decompose F into disjoint dialects into a partial order, a favorable situation that is proven in Proposition 4 in the Appendix Section VI-B.

If two messages really mean the same thing, then we may treat one, the other, or both as required. That is, Equation (1) will force some correlated messages to be in some dialect together. It may happen that those messages' correlations may only end up impacting one dialect, or if they are correlated with multiple other messages, they may impact other dialects as well.

This implies that dialects are ambiguous in a limited way, and this ambiguity is central to our approach. Instead of focusing initially on finding the best possible model, we permit there to be many possible decompositions into dialects. Furthermore, Theorems 1 and 2 in Section IV assert that this ambiguity is rather benign. In short, bounds on the number of dialects and the dialects themselves can be obtained algorithmically.

A. Conditionally independent mixtures with required messages

The *power set* 2^M of M consists of all subsets of M. Equation (1) can be thought of as defining a function $P(\cdot|A) : 2^M \to [0, 1]$ for each dialect A. We can think of each message pattern K as a binary sequence of length $\#M$. In that interpretation, each message $k \in M$ is a variable that can take the value 1 if the message is present in K or 0 if it is absent from K. With a slight abuse of notation, the expression for $P(K|A)$ can then be written

$$P(K|A) = \begin{cases} 0 \text{ if } R_A \not\subseteq K, \\ P(R_A|A) \prod_{k \notin R_A} P(k|A) \text{ otherwise,} \end{cases}$$

because $P(\{k = 0\}) = 1 - P(\{k = 1\})$.

The subset operation $\subseteq$ turns the power set into a partially ordered set $(2^M, \subseteq)$. Our methodology in Section IV applies to any partially ordered set, and so applies to message patterns. The partial ordering enforces certain ordering relationships for

the probabilities of messages patterns exhibited by files within a dialect.

In [1, Lem. 1], it was proven that if each message occurs with probability less than $1/2$, then the probabilities defined by Equation (1) decrease as more messages occur once the required messages have occurred.

The assumption that $P(\{k\}|A) < 1/2$ is necessary but benign. When a given message occurs on more than half of the files, we may simply instead consider the absence rather than the presence of that message.

Lemma 1. *Suppose that within a dialect A, $P(\{k\}|A) < 1/2$ for every message $k \in M$. The probability function defined by Equation* (1) *can also be written as*

$$P(K|A) = 1_{U_{R_A}}(K)g(K),$$

where

1) *R_A is the set of required messages for A,*
2) *$U_{R_A} = \{B \in 2^M : R_A \subseteq B\}$ is the support of the dialect A, that is the set of all message patterns containing R_A,*
3) *$1_{U_{R_A}}$ is the indicator function on U_{R_A}, and*
4) *$g : 2^M \to [0,1]$ is monotonic decreasing using the partial order $(2^M, \subseteq)$ on the domain and the usual ordering of the reals on the codomain.*

Proof. Equation (1) stipulates that if $k \in R_A$ but $k \notin K$, then $P(K|A) = 0$. This means that the support of $P(\cdot|A)$ is contained within the support of the indicator function on the set R_A of required messages.

On the other hand, since $P(\{k\}|A) < 1/2$ for every message $k \in M$, this implies that the probability decreases if we leave out a non-required message. This can be shown explicitly. Without loss of generality, suppose that K is in the support of the dialect, that $K = \{k_1, \dots, k_n\}$, and the required messages start at index $i+1$, so that $R_A = \{k_{i+1}, \dots\}$. Then we can write

$$\begin{aligned} P(k_1, \dots, k_n|A) &= P(k_1|A) \cdots P(k_i|A)P(k_{i+1} = 1, \dots|A) \\ &< P(k_1|A) \cdots P(k_{j-1}|A)P(k_{j+1}|A) \cdots \\ &\quad P(k_i|A)P(k_{i+1} = 1, \dots|A) \\ &< P(k_1, \dots, k_{j-1}, k_{j+1} \dots|A). \end{aligned}$$

Said another way, the probability is a monotonic decreasing function within U_{R_A}. □

If several disjoint dialects are present, the probability of message patterns being exhibited has a rather definite form.

Corollary 1. *Suppose that within a dialect A, $P(\{k\}|A) < 1/2$ for all messages k. Assuming dialects are disjoint, the joint distribution of messages over all files is then*

$$\begin{aligned} P(K) &= \sum_A P(K|A)P(A) \\ &= \sum_A 1_{U_{R_A}}(K)g_A(K), \end{aligned} \tag{2}$$

where $1_{U_{R_A}}$ and g_A are the functions defined in the statement of Lemma 1 associated to dialect A.

It is worth noting that the assumption of disjointness holds for some file formats but not others. For instance, CSV files can be reasonably supposed to exhibit disjoint dialects. If disjointness does not hold, we contend that Equation (2) may still be a useful model. Even though PDF and NITF files do not exhibit disjoint dialects, the model still provides useful information in our analysis in Section III.

Proposition 1. *Suppose that within a dialect A, $P(\{k\}|A) < 1/2$ for all messages k. Under the model given by Equation* (1)*, the support of a dialect has a unique minimal number of messages that occur, namely the required messages.*

Proof. According to Lemma 1, for a dialect A,

$$P(K|A) = 1_{U_{R_A}}(K)g(K)$$

for a monotonic decreasing g. This ensures that the support of the dialect is contained within U_{R_A}.

By way of contradiction, suppose that there were two minimal sets of messages that occur. This is equivalent to saying that there are at least two proper subsets $S_1 \subset R_A$ and $S_2 \subset R_A$ of the required messages R_A for which $P(S_1|A)$ and $P(S_2|A)$ are both nonzero, yet $P(R_A|A) = 0$. Notice that by construction,

$$P(S_1|A) = g(S_1),$$

and

$$P(R_A|A) = g(R).$$

We have just shown that $g(S_1) > g(R_A)$, yet this violates monotonicity and so is a contradiction. □

III. Experimental results

Proposition 1 yields a decomposition of the joint message probability into dialects specified by minimal sets of required messages. Assuming that these decompositions can be obtained—Section IV explains how to construct them—this section discusses how these decompositions can partition a given set of files into semantically useful dialects for further exploration by other means.

As a preprocessing stage, any messages that occurred on more than half of the files were inverted. That is, instead of noting the presence of these messages, we record their absence.

A. CSV

The humble comma separated value (CSV) file format appears at first glance to be completely defined by its name. It is a text file format for specifying tabular data, consisting of cells grouped into rows and columns. Each row corresponds to a line in the file, delimited by one of a handful of line ending characters. Each column is delimited by a comma character. This simple characterization quickly goes awry as what constitutes a "comma" varies with language and text file encoding [10]. Moreover, since cells might contain delimiters for line endings or commas, some kind of quoting is required. Again, quote characters vary with encoding. Finally, because CSV files are often consumed by spreadsheet applications,

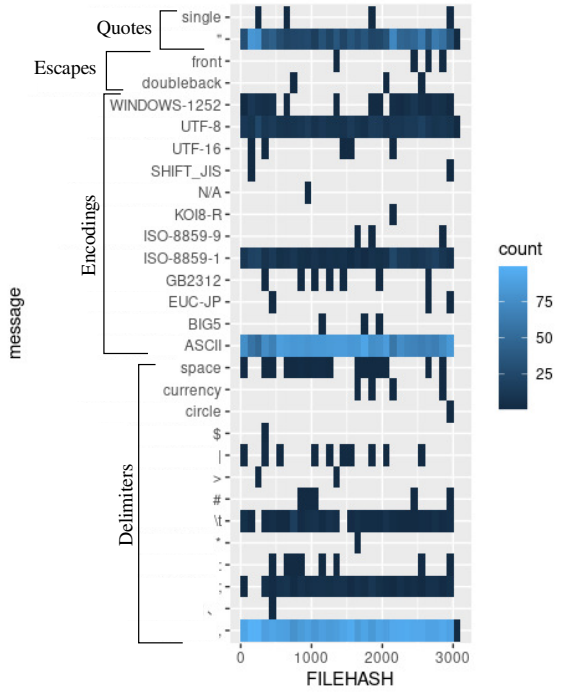


Fig. 1. Summary of our CSV data: Rows correspond to possible messages, while columns correspond to files within our sample. A gray cell indicates that the message did not occur for the corresponding file, whereas a shaded cell indicates that it occurred. Due to limitations of horizontal resolution, files are aggregated into bins in which counts are displayed.

they can contain formulae or fragments of executable code that can interact with the parser in surprising ways [16].

To demonstrate the dialects that are present within a corpus of CSV files, we drew a simple random sample of 3005 files from the dataset described in [10]. To each of these files, we extracted a total of 33 messages obtained by the `CleverCSV` tool described in the same article. In total, the messages consist of

- 14 delimiters,
- 3 quote characters,
- 3 escape characters, and
- 13 distinct text encodings.

Figure 1 shows a representation of the data obtained by this process. Each row corresponds to a distinct message, and each column corresponds to a distinct file. There are 29 rows in Figure 1 because 4 messages (1 quote character, 2 escape characters, and 1 delimiter) did not occur on any file in our sample.

The horizontal stripes in Figure 1 indicate that there are some messages that are very frequent (ASCII encoding and ASCII comma, for instance).

Message patterns are formed by considering files that elicit several messages simultaneously. Table I records the number of files exhibiting each message pattern. Each message pattern is a subset of the messages shown in Figure 1. Only message patterns for which 5 or more files were present are shown.

TABLE I
FILE COUNTS FOR THE MOST COMMON MESSAGE PATTERNS IN THE CSV DATA

File count	Message pattern
1417	`, ASCII`
682	`, " ASCII`
196	`, " UTF-8`
156	`, " ISO-8859-1`
119	`, UTF-8`
64	`, " WINDOWS-1252`
55	`; ISO-8859-1`
51	`TAB ASCII`
48	`TAB ISO-8859-1`
45	`; ASCII`
29	`ASCII`
18	`space ASCII`
18	`, ISO-8859-1`
11	`; " ASCII`
10	`; " UTF-8`
8	`| ASCII`
8	`, GB2312`
8	`; " ISO-8859-1`
7	`: ASCII`
6	`, WINDOWS-1252`

Most CSV files (about 89%) correspond to text files delimited with commas, which is hopefully not too surprising. ASCII text files delimited by ASCII commas correspond to about 70% of the total sample. About half of these ASCII files contain ASCII `"` as a quotation character. It is worth noting that the ASCII `TAB` character is often used as a field delimiter in Microsoft Excel.

The perspective arising from Table I is useful but does not do well for finding unusual dialects. As described in [1], a different summary of the file-message data is a partially ordered set in which each node corresponds to a message pattern, and the order relation connects pairs of message patterns obtained by adding additional messages. Figure 2 shows the Hasse diagram of this partial order for our data. In the figure, the size of each node is determined by the number of files exhibiting its corresponding message pattern. Because the number of files in each message pattern varies tremendously, the sizes are scaled logarithmically.

It is immediately apparent that the graph consists of many disconnected components, each of which correspond to at least one dialect. Many of these components correspond to different text encodings. As should be expected, the ASCII-encoded files form the largest component.

The statistical model described in Section II suggests that relationships between message patterns might account for their prevalence. For instance, consider the message pattern `TAB ASCII`. Given that 51 of these files occur, it should not be surprising that there are also some files that exhibit these two messages along with a quotation or escape character. Moreover, the statistical model suggests if these other files are not too common, then we should consider them to be part of the same dialect.

There is some ambiguity in how the dialects are formed, though Theorem 2 in Section VI-B establishes that there is

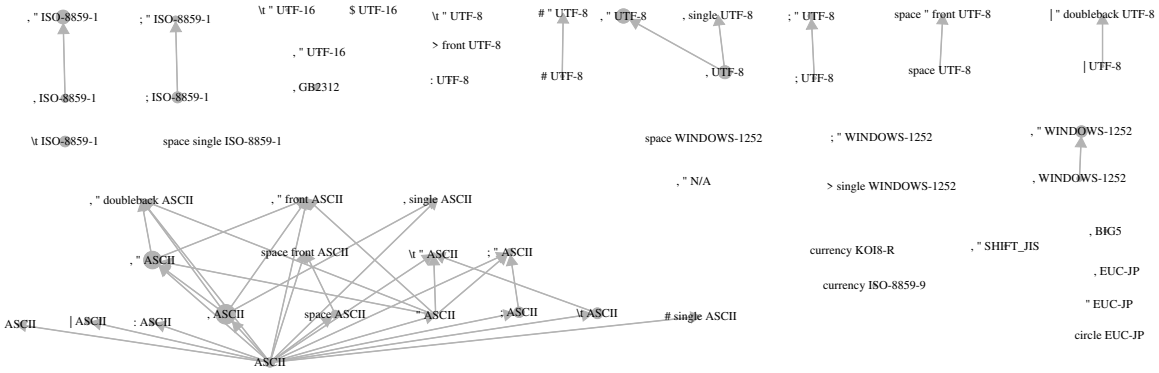


Fig. 2. Partial order of message patterns in the CSV data. Only those message patterns present in the data are shown. The size of each node is logarithmic in the number of files exhibiting the corresponding message pattern.

TABLE II
LARGEST CSV CANDIDATE DIALECTS, IN ORDER OF DISCOVERY

File count at root	Required messages
1388	`, ASCII`
119	`, UTF-8`
77	`, " UTF-8`
18	`, ISO-8859-1`
7	`; " UTF-8`
22	`TAB ASCII`
138	`, " ISO-8859-1`
48	`TAB ISO-8859-1`
58	`, " WINDOWS-1252`
29	`ASCII`
55	`; ISO-8859-1`
16	`; ASCII`
8	`, GB2312`
6	`, WINDOWS-1252`

TABLE III
MESSAGES ASSOCIATED TO EACH PDF PARSER

Parser	Message count
`caradoc`	253
`hammer`	65
`mutool`	796
`origami`	39
`pdfium`	21
`pdfminer`	62
`pdftk`	29
`pdftools`	10
`poppler`	995
`qpdf`	136
`xpdf`	598
Total	3004

a unique set of dialects that are the largest. Taking that as a given, the resulting dialects we obtain are shown in Table II.

The "File count at root" column reports the number of files exhibiting the required messages *and no others*. According to Lemma 1, the number of files in the dialect associated to any other message pattern will not exceed this value. In terms of file count, the top entries in Table II are the comma-delimited text files with various text encodings. The number of dialects with at least 5 files is 14, which is smaller than the 20 message patterns with at least 5 files, so some consolidation of the data has occurred. The `, " ASCII` message pattern has been subsumed into `, ASCII`, for instance. The numbers of files in each have decreased somewhat because the statistical model expects a few files exhibiting message pattern `, ASCII` to arise as part of the `ASCII` dialect. In short, the model expects that there are probably a few files that exhibited message pattern `, ASCII` that are not actually CSV files—tabular data—but are instead unstructured ASCII text files containing commas.

B. PDF

The Portable Document File (PDF) format is defined by the ISO 32000-2 standard. For this exercise, we used a sample of 10000 files curated by the Test and Evaluation Team for the DARPA SafeDocs evaluation exercise 4.

Each file was processed through 11 distinct parsers, run with various options. A total of 3004 Boolean messages were collected, as shown in Table III. One message per parser is an exit code corresponding to the presence of an error. The rest of the messages correspond to specific regular expressions (regexes) run against `stderr` and `stdout`, as explained previously in [1]. Several of these messages were found to play an important role in identifying dialects, and appear in Table IV.

After processing, we found 1658 distinct message patterns, of which those with file count of at least 100 are shown in Table V. It should be noted that valid files often still produce numerous warnings and other output.

The proof of Proposition 3 compresses the message patterns into 4 dialects with root file count greater than 100, as shown in Table VI. These four dialects have a fairly clear interpretation, as shown in the last column. The latter two dialects appear to correspond to different kinds of syntax errors.

As a comparison, if we consider file counts of at least 25, we found 10 dialects with root file count greater than 25 compared to 43 message patterns. A format analyst need only consider about one-quarter as many dialects as overall message patterns.

TABLE IV
SAMPLE MESSAGES IN OUR PDF DATA RELEVANT FOR CANDIDATE DIALECTS

Message	parser	regex
69	caradoc	PDF error : Syntax error at offset \d+ \[0x[A-Fa-f\d]+\] in file !
163	caradoc	PDF error : Syntax error at offset .* in file !
217	caradoc	PDF error : Lexing error : unexpected character : 0x[A-Fa-f\d]+ at offset…
220	caradoc	PDF error : Lexing error : unexpected word at offset \d+ \[0x[A-Fa-f\d]+\…
250	caradoc	Warning : Flate\/Zlib stream with appended newline in object .*
96,188,251	caradoc	Exit code meaning error
255	hammer	.*: no parse
258	hammer	(?:/[a-zA-Z\d \-]+)+/[A-Fa-f\d]+: error after position \d+ \(0x[A-Fa-f\d…
297	hammer	VIOLATION … No newline before 'endstream' …
308	hammer	VIOLATION … Missing endobj token \(seve…
313	hammer	VIOLATION … No linefeed after 'stream' \…
314	hammer	VIOLATION … Nonconformant WS at end of x…
316	hammer	Exit code meaning error
482	mutool	warning: line feed missing after stream begin marker \(\d+ \d+ R\)
720	mutool	warning: line feed missing after stream begin marker \(\d+ \d+ R\)
899	mutool	page (?:/[a-zA-Z\d]+)+/[A-Fa-f\]+ \d +
978	mutool	warning: line feed missing after stream begin marker \(\d+ \d+ R\)
1143	origami	.*Object shall end with 'endobj' statement.*
1153	origami	Exit code meaning error
2346	qpdf	WARNING: .*: expected endobj
2384	qpdf	WARNING: .*: stream keyword followed by carriage return only
2889	xpdf	Syntax Warning.*: Substituting font '.*' for '.*'
3015	xpdf	non_embedded_font

TABLE V
FILE COUNTS FOR THE MOST COMMON MESSAGE PATTERNS IN THE PDF DATA

File count	Message pattern
3761	250 251 899 1153
347	251 297 899 1153
319	250 251 899 1153 2888
186	217 251 899 1153
117	92 96 184 188 247 251 899 1153
116	200 250 251 899 1153
114	69 96 163 188 220 251 255 258 297 308 313 ...
107	234 251 899 1153
102	228 251 899 1153
101	7 96 104 188 217 251 899 1153

TABLE VI
LARGEST PDF CANDIDATE DIALECTS

File count at root	Required messages	Interpretation
3684	250 251 899 1153	Compressed stream error
270	251 297 899 1153	Missing/misplaced endstream delimiter
111	69 96 163 188 220 251 255 258 297 308 313 ...	Syntax error
109	217 251 899 1153	Syntax error

C. NITF

The National Imagery Transmission Format (NITF) is used by many US Government entities to share geocoded imagery. These files contain a complicated header and a data payload. There are several distinct NITF parsers in wide usage, which has lead to some format divergence.

As part of the DARPA SafeDocs hackathon exercise 5, the Test and Evaluation Team provided a set of 2626 NITF files. Against each of these files, 6 parsers were run. The output of stderr, stdout were collected along with the parser's return code. Using the same process described in [1] for PDF, suitably modified for NITF, a collection of regexes were run on this output to produce 136 possible messages. The breakdown of parsers and messages is shown in Figure VII.

TABLE VII
PARSERS USED TO PROCESS THE NITF DATA

Parser	Message count
afrl	42
codice	28
gdal	36
hammer_nitf	15
kaitai	6
nitro	9
Total	136

The most common messages are shown in Table VIII. Several regular expressions matched more than 50% of the time. Since this violates the assumptions on message probability in Section II-A, we used the absence of a match in what follows.

In our data, 103 of the messages are extant. There are 20 distinct message patterns with file count at least 25, which are shown in Table IX.

The decomposition of the data into dialects is summarized in Table X. The "File count at root" column reports the number of files exhibiting the required messages *and no others*. According to Lemma 1, the number of files in the dialect associated to any other message pattern will not exceed this value.

There are 12 dialects with required messages having a file count at least 25, as shown in Table X. This is an improvement over the 20 message patterns that Table IX presents. Referring back to Table VIII, all but the two most common dialects

TABLE VIII
MOST COMMON MESSAGES IN THE NITF DATA

Message	File count	parser	regex
59	1051	codice	Absence of `Parse error\n`
102	1039	gdal	Absence of `gdalinfo failed \- unable to open '.*'\.`
107	1038	hammer_nitf	Absence of errors in exit code
71	1029	gdal	Absence of errors in exit code
1	825	afrl	Absence of errors in exit code
37	812	afrl	`Error reading, read returned .*\. \(start = .*, …`
94	527	gdal	`ERROR \d+: Not enough bytes to read segment info`
108	470	hammer_nitf	`/[a-zA-Z\d _\\.\-\(\):/,+]+\.[a-zA-Z\d]+: no parse`
113	420	hammer_nitf	`VIOLATION … Invalid file length in header \(severity=\d+\)`
21	394	afrl	`Error reading header.*`
12	308	afrl	`user defined data length = \d+`
103	241	gdal	`gdal ERROR .*: NITF Header Length \(.*\) seems…`
119	241	hammer_nitf	`VIOLATION … Invalid number of graph segments \(severity=\d+\)`
99	227	gdal	`Warning \d+: … appears to be an NITF file, but no image …`

TABLE IX
FILE COUNTS FOR THE MOST COMMON MESSAGE PATTERNS IN THE NITF DATA

File count	Message pattern
357	1 59 71 102 107
151	1 12 59 71 102 107
100	1 59 71 99 102 107 122
70	14 23 94
70	94
66	21 37 81 113
59	1 59 71 99 102 107
56	103 113
50	15 37 86
48	21 37 94 113
47	33 37 103 113
44	22 37 76 108 119
43	94 111
42	21 37 103 113
30	21 37 94 119
29	17 59 71 102 107
29	21 37 76 113
29	7 59 71 102 107
28	12 34 37 40 94
26	2 12 59 79 82 107

TABLE X
LARGEST NITF CANDIDATE DIALECTS

File count at root	Required messages	Interpretation
352	1 59 71 102 107	Valid files
93	1 59 71 99 102 107 122	Corrupted data payload
70	94	Read access error
60	14 23 94	Read access error
54	103 113	Corrupted header length
49	15 37 86	Read access error
43	21 37 81 113	Corrupted header length
41	21 37 94 113	Corrupted header length
27	22 37 76 108 119	Read access error
26	21 37 94 119	Corrupted header
26	2 12 59 79 82 107	Valid but unsupported version
25	21 37 76 113	Corrupted header length

shown contains some kind of parser *error* message.

While not every possible message corresponds to a violation of length fields, most of the dialects shown in Table X correspond to a corrupted length field within the NITF header. Because the NITF specification permits random access to the data payload, length field corruption explains the presence of messages 37 and 94, which indicate reading beyond the end of the file. What ultimately distinguishes the dialects is which length fields were corrupted. Due to differences in how the parsers operate, different fields are collected at different points in the parse by different parsers.

Further examination of the files in the second-to-last dialect in Table X, the one that contains message 2, revealed that these files were for a version not supported by the `afrl` parser.

IV. DETAILED METHODOLOGY

Our goal is to find candidate dialects from the probabilities $P(K)$ of each message pattern. We show in Proposition 3 that there are decompositions of these data into a mixture of disjoint dialects of the form postulated in Corollary 1. The proof of Proposition 3 is sufficiently constructive that it can be taken as defining an algorithm that finds a decomposition by greedily selecting the largest dialects first.

Although our probabilistic model is posed over the power set of messages partially ordered by subset $(2^M, \subseteq)$, we will instead establish results for an arbitrary partially ordered set X. This can yield a substantial savings in memory usage and runtime of any algorithms working on the data, because we need only consider those message patterns that are actually present in the data. Moreover, at the level of generality used, our data could be formatted as probabilities taking values between 0 and 1 or equally well as counts of files.

As suggested in Section II-A, the data are formatted as a function $f : X \to [0, \infty)$ from a partially ordered set $(X, \leq)$ to the nonnegative real numbers. We begin by relaxing from independent mixtures (Corollary 1) to a decomposition into monotonic functions.

Definition 2. Suppose that $f : X \to [0, \infty)$ is a function from a finite partially ordered set $(X, \leq)$ to the nonnegative real numbers. A *monotonic decomposition* expresses f as a sum of functions

$$f(x) = \sum_{k=1}^{N} 1_{U_{y_k}}(x) g_k(x) \tag{3}$$

where $g_k : X \to [0, \infty)$ is monotonic decreasing, and $U_{y_k} = \{x \in X : x \geq y_k\}$ is an upwardly closed set. Through a slight abuse of naming, which is meant to evoke the usage for files, we will call each term in the above sum a *dialect*. Each y_k is called a *root node*, and plays the role of the pattern of required messages for the dialect.

Duplicate y_k and/or g_k are permissible in a monotonic decomposition.

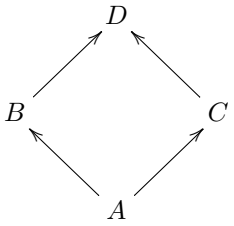
Monotonic decompositions generalize the probabilistic model posited in Section II, in which the values of f could correspond to probabilities or file counts of message patterns.

Proposition 2. *Suppose that there is a set of messages for which the probability each message pattern in each dialect is given by Equation* (1)*. Assume that dialects are disjoint so that Corollary 1 applies. Then Equation* (2) *is a monotonic decomposition of the joint probability distribution over all message patterns.*

Proof. According to Lemma 1, each dialect corresponds to a term of the form $1_{U_y} g(x)$ where g is a monotonic decreasing function. According to Corollary 1, the formula for the joint probability distribution for all messages patterns is of precisely the same form as required by Equation (3) in Definition 2. □

Nonuniqueness is a general feature of monotonic decompositions, and is not due to any particular algorithm for finding them.

Example 1. Consider the partially ordered set given by



with the function f given by

$$f(A) := 0,\ f(B) := 4,\ f(C) := 4,\ \text{and } f(D) := 5.$$

One can interpret these values as counts of files for four different message patterns. In this case B and C represent two distinct, but overlapping message patterns. A represents the message pattern consisting of the intersection of the patterns for B and C. Finally, D represents the message pattern consisting of the union of the patterns for B and C.

The function f can be written as the sum

$$f = 1_{U_B} g_1 + 1_{U_C} g_2,$$

where

$$g_1(A) := 5,\ g_1(B) := 4,\ g_1(C) := 4,\ \text{and } g_1(D) := 2,$$

and

$$g_2(A) := 5,\ g_2(B) := 4,\ g_2(C) := 4,\ \text{and } g_2(D) := 3.$$

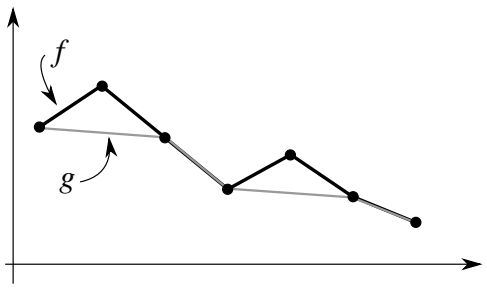


Fig. 3. An example of the function g guaranteed by Lemma 3

Evidently, both g_1 and g_2 are monotonic decreasing. However, it is also true that

$$f = 1_{U_B} g_2 + 1_{U_C} g_1,$$

which contradicts the uniqueness of the decomposition, since exactly the same two open sets are used.

Our experimental results follow from a constructive proof of the following.

Proposition 3. *Every function $f : X \to [0, \infty)$ from a finite partially ordered set $(X, \leq)$ to the nonnegative real numbers has a monotonic decomposition.*

We will prove Proposition 3 constructively (algorithmically after certain choices are made) later in this section, after establishing two Lemmas as tools.

Lemma 2. *Suppose that $f : X \to [0, \infty)$ is a function from a partially ordered set $(X, \leq)$ to the nonnegative real numbers. If $g_1, g_2 : X \to [0, \infty)$ are two functions satisfying*

1) *both g_i are* monotonic decreasing*: if $x \leq y$ are two elements of X, then $g_i(x) \geq g_i(y)$, and*
2) *$g_i(x) \leq f(x)$ for all $x \in X$ and both g_i.*

Then the function

$$h(x) := \max\{g_1(x), g_2(x)\}$$

satisfies the same two conditions.

Proof. The fact that $h(x) \leq f(x)$ for all $x \in X$ follows immediately from the fact that $h(x)$ is equal to either $g_1(x)$, or $g_2(x)$, or both. Now suppose that $x \leq y$ are two elements of X. Without loss of generality, suppose that $h(x) = g_1(x)$. This means that $g_1(x) \geq g_2(x)$, and by assumption $g_2(x) \geq g_2(y)$. By transitivity, this means that $h(x) = g_1(x) \geq g_2(y)$. By assumption, we have that $h(x) = g_1(x) \geq g_1(y)$ as well. Since $h(y)$ is equal to the larger of $g_1(y)$ and $g_2(y)$, this means that $h(x)$ is larger than $h(y)$. □

Lemma 3. *Suppose that $f : X \to [0, \infty)$ is a function from a finite partially ordered set $(X, \leq)$ to the nonnegative real numbers. The set of monotonic decreasing functions $g : X \to [0, \infty)$ such that $g(x) \leq f(x)$ for all $x \in X$ has a unique maximal element. (See Figure 3.)*

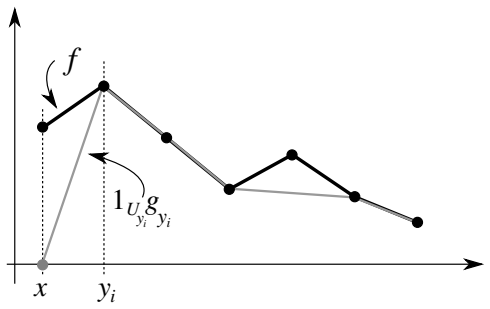


Fig. 4. Construction of the function g_{y_i} in the proof of Proposition 3

Proof. All we need to show is that the set in question is nonempty, since Lemma 2 does the rest. This is easy because the zero function is in the set. □

There is a greedy, recursive algorithm that constructs the function g guaranteed by Lemma 3.

Algorithm 1. 1) *Start with the set of minimal elements M_0 of X.*
2) *We can without any trouble define $g(m) := f(m)$ for every $m \in M$, as this is evidently maximal in all cases.*
3) *In preparation for the recursive step, let $L_0 := M_0$.*
4) *For the recursive step, assume g is already defined on some lower-closed subset L_k of X.*
   a) *Consider the set M_k of minimal elements of $X \setminus L_k$.*
   b) *For each $m \in M_k$, define $g(m) := \min\{f(m)\} \cup \{g(x) : x < m\}$, noting that every x in the latter set is an element of L_k so $g(x)$ is well-defined. Defining g in this way ensures that it is upper bounded by f, is monotonic decreasing, yet is otherwise maximal.*
   c) *In preparation for the next recursive step, let $L_{k+1} := L_k \cup M_k$.*

The greediness of Algorithm 1 makes it scale linearly in the number of relations in X.

These maximal monotonic decreasing functions can be used to decompose an arbitrary function into a sum of monotonic decreasing functions whose domains are restricted appropriately.

Proof. (of Proposition 3) Proceed by induction on the number of places where f fails to be monotonic decreasing.

- Base case: f is monotonic decreasing. If there is one minimal element, y of X, then we merely take $y_1 := y$, and let $g_1 := f$. However, if there is more than one minimal element, then things become annoyingly non-unique. This can be resolved in various ways, for instance using the following tie-breaking procedure. Let $y_1, \ldots, y_m$ be an arbitrary ordering of the minimal elements of X. Since they are all minimal, they are mutually incomparable elements of X. Define

$$g_i(x) := 1_{A_i}(x) f(x),$$

where

$$A_i = U_{y_i} \setminus \bigcup_{j=1}^{i-1} U_{y_j}.$$

Notice that since each 1_{A_i} is monotonic decreasing—it is 1 on y_i, but eventually drops to 0 on sufficiently large elements of X—the resulting g_i functions are also monotonic decreasing. Moreover, by construction each element x of X is an element of exactly one A_i set. Therefore, the decomposition formula for f holds.

- Induction case: Suppose that f is not monotonic decreasing at k elements $\{y_1, \ldots, y_k\}$ of X. For each of these elements y_i, there is an x in X with $x \leq y_i$ but $f(x) < f(y_i)$. At least one of these y_i is minimal among the set $\{y_1, \ldots, y_k\}$, which means that for each $j \neq i$, either $y_i < y_j$ or y_i and y_j are incomparable. Use Lemma 3 to construct a maximal monotonic decreasing function g_{y_i} on U_{y_i} that is bounded above by f. We will show that the residual function $f - 1_{U_{y_i}} g_{y_i}$ fails to be monotonic at not more than $k-1$ elements of X.
We assumed that there was an x such that $f(x) < f(y_i)$. Due to the hypothesis that g_{y_i} is bounded above by f, this means that $g_{y_i}(y_i) \leq f(y_i)$. On the other hand, given that Lemma 3 asserts a maximal such g_{y_i} exists on U_{y_i}, we must conclude that $g_{y_i}(y_i) = f(y_i)$, as shown in Figure 4, since that value has no further impact on the monotonicity of g_{y_i}. Therefore, the residual function $f - 1_{U_{y_i}} g_{y_i}$ takes the value 0 on y_i, and therefore automatically satisfies

$$\begin{aligned} f(x) &= f(x) - 1_{U_{y_i}}(x) g_{y_i}(x) \\ &\geq 0 = f(y_i) - 1_{U_{y_i}}(y_i) g_{y_i}(y_i). \end{aligned}$$

Therefore, at least one violation of monotonicity in f is not present in the residual.
Let us establish that no new violations of monotonicity occur in the residual. Suppose that $x \leq z$ are two elements of X for which $f(x) \geq f(z)$. If both elements are outside U_{y_i}, then the residual is unchanged from f on these two elements. If x is outside U_{y_i} but $z \in U_{y_i}$, then

$$\begin{aligned} f(x) - 1_{U_{y_i}}(x) g_{y_i}(x) &= f(x) \\ &\geq f(z) \\ &\geq f(z) - 1_{U_{y_i}}(z) g_{y_i}(z), \end{aligned}$$

since g_{y_i} is nonnegative by construction. Finally, assume that both x (and therefore z) are elements of U_{y_i} and that $f(x) \geq f(z)$. While an arbitrary monotonic decreasing function h bounded above by f might not result in $f(x) - h(x) \geq f(z) - h(z)$, this cannot happen with g_{y_i} due to its maximality. We establish this by way of contradiction; suppose that

$$f(x) - g_{y_i}(x) < f(z) - g_{y_i}(z).$$

Rearranging this inequality yields

$$0 \leq f(x) - f(z) < g_{y_i}(x) - g_{y_i}(z),$$

which means that there is a monotonic decreasing h with $h(x) = g_{y_i}(x)$ and $g_{y_i}(z) < h(z) \leq f(z)$, contradicting the maximality of g_{y_i}. We have therefore established that the residual $f - 1_{U_{y_i}} g_{y_i}$ has strictly fewer violations of monotonicity than f.

□

Example 2. Let us apply Algorithm 1 repeatedly to Example 1, according to the recipe in the proof of Proposition 3 to show how dialects are collected. We need to run Algorithm 1 for each dialect, which will yield a new g function for each dialect. We will call these functions g_A, g_B, g_C, and g_D.

The minimal element of the partial order is A, so we set $M_0 = \{A\}$, leading to $g_A(A) := f(A) = 0$, and $L_0 = \{A\}$. Evidently, this means that the first dialect has a root node of A, but has all zero values since they must be bounded above by $g_A(A) = 0$.

To obtain the next dialect, we remove A from consideration, leaving $\{B, C, D\}$. There are two minimal elements, namely B and C. Let us consider B. We thus can take $g_B(B) := f(B) = 4$, and after one recursive step, we have $g_B(D) := 4$ after Step 4(b) in Algorithm 1.

Removing B from consideration, the minimal element of the resulting set $\{C, D\}$ is C, which will lead us to construct g_C. We can take $g_C(C) := f(C) = 4$. Since we want to result in a monotonic decomposition, we must have that

$$5 = f(D) = g_A(D) + g_B(D) + g_C(D) = 0 + 4 + g_C(D).$$

Thus $g_C(D) = 1$. At this point, the residual in Equation (3) is zero, so there is nothing left to do.

This completes the monotonic decomposition, namely that

$$f = 1_{U_A} g_A + 1_{U_B} g_B + 1_{U_C} g_C, \tag{4}$$

where

$$g_A(A) := 0,\ g_A(B) := 0,\ g_A(C) := 0,\ \text{and}\ g_A(D) := 0,$$

$$g_B(B) := 4,\ g_B(C) := 0,\ \text{and}\ g_B(D) := 4,$$

and

$$g_C(B) := 0,\ g_C(C) := 4,\ \text{and}\ g_C(D) := 1.$$

Notice that we never defined $g_B(A)$ or $g_C(A)$. While strictly speaking this does not fit the form of Equation (3), these values lie outside the support of their respective dialects. We can easily fill the missing values with the maximum value of each function to comply with Equation (3). Additionally, since g_A is identically zero, we may remove it from further consideration.

The decomposition proposed in the proof of Proposition 3 is not unique. This means that there are sometimes different possible choices of dialects that result in the same message probabilities. Fortunately, the collection of dialects produced by the proof of Proposition 3 is the coarsest collection that is consistent with the specified probabilities.

Theorem 1. *The procedure defined in the proof of Proposition 3, yields a minimally refined monotonic decomposition for an arbitrary nonnegative function f on a finite partially ordered set.*

Assuming that the probabilities of the true dialects are given by an independent mixture model, then the support of each dialect is a subset of a dialect found by Algorithm 1.

Theorem 2. *The number of dialects is bounded below by the number of support sets in the irredundant cover of X induced by any minimal irredundant monotonic decomposition of the joint probability distribution function.*

Hence, the decomposition constructed by the proof of Proposition 3 yields a bound on the number and structure of dialects. In particular, the sets of required messages found by this procedure correspond to those of some true dialects, though there may be other dialects that remain to be found. Theorems 1 and 2 are proven in the Appendix.

V. Conclusion

This paper presented a novel statistically-based method for partitioning sets of files into format dialects based upon their behaviors when parsed. As a direct consequence of Theorems 1 and 2, our method yields the coarsest such partition that could be consistent with the statistical model. This means that a format analyst can begin their analysis with a minimal number of dialects. In practice, an analyst needs to consider about half the number of dialects as distinct message patterns. Intuitively, this considerably reduces their cognitive load when studying a complex format.

Acknowledgments

The authors would like to thank the SafeDocs test and evaluation team, including NASA (National Aeronautics and Space Administration) Jet Propulsion Laboratory, California Institute of Technology and the PDF Association, Inc., for providing the test data. The authors would like to thank Cory Anderson for the initial processing of the files into sets of messages.

This material is based upon work supported by the Defense Advanced Research Projects Agency (DARPA) SafeDocs program under contract HR001119C0072. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of DARPA.

Conflict of interest

The authors state that there is no conflict of interest.

References

[1] M. Robinson, L. W. Li, C. Anderson, and S. Huntsman, "Statistical detection of format dialects using the weighted Dowker complex," in *2022 IEEE Security and Privacy Workshops (SPW)*, 2022, pp. 98–112. [Online]. Available: http://arxiv.org/pdf/2201.08267

[2] D. Scofield, C. Miles, and S. Kuhn, "Fast model learning for the detection of malicious digital documents," in *SSPREW-7*, December 2017.

[3] M. Robinson, "Looking for non-compliant documents using error messages from multiple parsers," in *2021 IEEE Security and Privacy Workshops (SPW)*, 2021, pp. 184–193. [Online]. Available: https://doi.org/10.1109/SPW53761.2021.00032

[4] K. Ambrose, S. Huntsman, M. Robinson, and M. Yutin, "Topological differential testing, `arxiv:2003.00976`," 2020. [Online]. Available: https://arxiv.org/abs/2003.00976

[5] M. Belaoued and S. Mazouzi, "A real-time PE-malware detection system based on chi-square test and PE-file features," in *IFIP International Conference on Computer Science and its Applications*. Springer, 2015, pp. 416–425.

[6] B. A. S. Al-rimy, M. A. Maarof, and S. Z. M. Shaid, "Ransomware threat success factors, taxonomy, and countermeasures: A survey and research directions," *Computers & Security*, vol. 74, pp. 144–166, 2018.

[7] S. D. S.L and J. CD, "Windows malware detector using convolutional neural network based on visualization images," *IEEE Transactions on Emerging Topics in Computing*, pp. 1–1, 2019. [Online]. Available: https://doi.org/10.1109/TETC.2019.2910086

[8] M. Alazab, "Profiling and classifying the behavior of malicious codes," *Journal of Systems and Software*, vol. 100, pp. 91 – 102, 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0164121214002283

[9] J. Demme, M. Maycock, J. Schmitz, A. Tang, A. Waksman, S. Sethumadhavan, and S. Stolfo, "On the feasibility of online malware detection with performance counters," *ACM SIGARCH Computer Architecture News*, vol. 41, no. 3, pp. 559–570, 2013.

[10] G. J. J. van den Burg, A. Nazábal, and C. Sutton, "Wrangling messy CSV files by detecting row and type patterns," *Data Mining and Knowledge Discovery*, vol. 33, no. 6, pp. 1799–1820, 2019. [Online]. Available: https://doi.org/10.1007/s10618-019-00646-y

[11] J. Marin, K. Mengersen, and C. P. Robert, "Bayesian modelling and inference on mixtures of distributions," in *Essential Bayesian models. Handbook of statistics: Bayesian thinking - modeling and computation. Vol. 25*, D. Dey and C. Rao, Eds. Elsevier, 2011).

[12] G. McLachlan and D. Peel, *Finite Mixture Models*. Wiley, 2000.

[13] A. Björner, "Topological methods," *Handbook of combinatorics*, vol. 2, pp. 1819–1872, 1995.

[14] M. Robinson, "Cosheaf representations of relations and Dowker complexes," *Journal of Applied and Computational Topology*, 2021. [Online]. Available: https://doi.org/10.1007/s41468-021-00078-y

[15] M. Brun and L. M. Salbu, "The rectangle complex of a relation," *Mediterranean Journal of Mathematics*, vol. 20, no. 1, pp. 1–8, 2023.

[16] J. Dileo, "CSV injection, RFC5322," March 2019. [Online]. Available: https://unpack.debug.su/pocorgtfo/pocorgtfo19.pdf

[17] C. R. Kime, R. P. Batni, and J. D. Russell, "An efficient algorithm for finding an irredundant set cover," *J. ACM*, vol. 21, no. 3, p. 351–355, jul 1974. [Online]. Available: https://doi.org/10.1145/321832.321833

[18] S.-i. Minato, "Fast generation of prime-irredundant covers from binary decision diagrams," *IEICE transactions on fundamentals of electronics, communications and computer sciences*, vol. 76, no. 6, pp. 967–973, 1993.

VI. Appendix

Although Proposition 3 constructs a monotonic decomposition of a functions, nonuniqueness can impede its practical utility. Theorem 1 asserts that the decomposition constructed in the proof of Proposition 3 is minimal in the sense that it cannot be decomposed further. Furthermore, Theorem 2 asserts that the decomposition so constructed provides an unambiguous lower bound on the true number of dialects. Theorem 1 is proven in Section VI-A. Theorem 2 is proven in Section VI-B.

A. Bounding the structure of dialects

An effective way to handle the ambiguity present among possible dialect decompositions is simply to embrace it. Some decompositions are evidently finer, in that they split the set of files into smaller dialects. This situation is easily characterized by the notion of *refinement*.

Definition 3. Suppose that a function $f : X \to [0, \infty)$ from a finite partially ordered set $(X, \leq)$ has multiple monotonic decompositions. We will say that the monotonic decomposition

$$f(x) = \sum_{k=1}^{N} 1_{U_{z_k}}(x) h_k(x)$$

refines the monotonic decomposition

$$f(x) = \sum_{j=1}^{M} 1_{U_{y_j}}(x) g_j(x)$$

if for every $k = 1, \cdots, N$, there is a j such that

1) $U_{z_k} \subseteq U_{y_j}$, and
2) $h_k(x) \leq g_j(x)$ for every $x \in U_{z_k}$.

Lemma 4. *The set of monotonic decompositions of a function $f : X \to [0, \infty)$ from a finite partially ordered set $(X, \leq)$ is itself a preordered set under refinement.*

Proof. This is merely a straightforward verification of the axioms. Suppose that we have three monotonic decompositions of f,

$$\begin{aligned} f(x) &= \sum_{k=1}^{N} 1_{U_{z_k}}(x) h_k(x) \\ &= \sum_{j=1}^{M} 1_{U_{y_j}}(x) g_j(x) \\ &= \sum_{\ell=1}^{Q} 1_{U_{w_\ell}}(x) p_\ell(x). \end{aligned}$$

- Reflexivity is trivial; both $\subseteq$ for sets and $\leq$ for functions are reflexive.
- For transitivity, suppose that the first monotonic decomposition refines the second, and that the second refines the third. For every $k = 1, \cdots, N$, there is a j such that $U_{z_k} \subseteq U_{y_j}$. Yet there is also an ℓ such that $U_{y_j} \subseteq U_{w_\ell}$. Thus $U_{z_k} \subseteq U_{w_\ell}$. For exactly these same indices, we have that
$$h_k(x) \leq g_j(x) \leq p_\ell(x)$$
for all $x \in U_{z_k}$. Hence, the first monotonic decomposition refines the third. □

We seek a monotonic decomposition that is *minimally refined*, there is no other monotonic decomposition refined by it. Because of cases like Example 1, minimally refined monotonic decompositions are not unique. In Example 1, both decompositions are minimally refined and neither refines the other.

The remainder of this section is devoted to the proof of Theorem 1. Before attempting the proof, it helps to consider the case of monotonic functions before considering general monotonic decompositions.

Lemma 5. *Suppose that $f : X \to [0, \infty)$ is a monotonic function from a finite partially ordered set $(X, \leq)$. If $(X, \leq)$ has a unique minimal element p, then every monotonic decomposition of f refines*
$$f = 1_{U_p} f. \tag{5}$$
Additionally, this monotonic decomposition refines no other monotonic decomposition of f.

Proof. For the first statement, suppose that
$$f(x) = \sum_{k=1}^{N} 1_{U_{z_k}}(x) h_k(x).$$
Evidently, since p is the minimal element of $(X, \leq)$, it follows that $U_{z_k} \subseteq 1_{U_p} = X$. Moreover, since $h_k(x)$ is nonnegative and the collection sums to f, it follows that $h_k \leq f$.

Conversely, the only way that the monotonic decomposition defined by Equation (5) refines any other is that $z_k = p$ for some k. If this is the case, refinement requires that $f \leq h_k$. However, since the h_k must sum to f, is still the case that $h_k \leq f$. Hence $h_k = f$. □

Proof. (of Theorem 1) Suppose that $f : X \to [0, \infty)$ is a function from a finite partially ordered set $(X, \leq)$.

Lemma 5 can be used with Lemma 3 to rule out refinement by certain decompositions supported on minimal elements. Suppose tentatively that $p \in X$ is a minimal element of $(X, \leq)$. If $g_p : U_p \to [0, \infty)$ is the unique maximal monotonic decreasing function such that $g \leq f$ guaranteed by Lemma 3, then
$$f = 1_{U_p} g_p + \sum_{k=1}^{N} 1_{U_{z_k}}(x) h_k(x)$$
cannot refine any monotonic decomposition which does not also contain the term $1_{U_p} g_p$.

With this situation treated, now consider the case where there are multiple minimal elements of $(X, \leq)$. Let $p_1, \ldots, p_m$ be all of the minimal elements of a finite partially ordered set $(X, \leq)$.

Define monotonic decreasing functions $g_i : X \to [0, \infty)$ for each $i = 1, \ldots, m$ inductively via

- Base case: g_1 is the unique maximal monotonic decreasing function on U_{p_1} such that $g \leq f$ guaranteed by Lemma 3,
- Induction case: g_i is the unique maximal monotonic decreasing function on U_{p_1} such that $g \leq \left(f - \sum_{j=1}^{i} 1_{U_{p_i}} g_i\right)$ guaranteed by Lemma 3.

Any monotonic decomposition of the form
$$f(x) = \sum_{j=1}^{m} 1_{U_{p_i}} g_i(x) + \sum_{k=1}^{N} 1_{U_{y_k}} h_k(x)$$
cannot refine any monotonic decomposition not containing all of the terms in the first sum.

The reader is cautioned that changing the ordering of the p_i in the above construction will generally yield different corresponding g_i functions. The monotonic decompositions so arising cannot refine each other as a result. The tie-breaking procedure used in the proof of Proposition 3 provides one such option for an ordering.

The full statement of the Theorem follows by mimicking the induction case of the proof of Proposition 3. That is, we repeat the above procedure with $\left(f - \sum_{j=1}^{m} 1_{U_{p_i}} g_i\right)$ instead of f, and restrict the domain to $X \setminus \{p_1, \ldots, p_m\}$. At each iteration, we obtain more terms of the minimally refined monotonic decomposition. □

B. The structure of the refinement preorder

In the previous section, it was shown (Lemma 4) that monotonic decompositions are preordered by refinement. We now establish that this preorder can be strengthened to a partial order (Proposition 4) if redundancies of a certain kind are eliminated.

Lemma 6. *Suppose that there are two monotonic decompositions of a function $f : X \to [0, \infty)$ that refine each other, and that these two decompositions can be written as*
$$f(x) = \sum_{k=1}^{N} 1_{U_{z_k}}(x) h_k(x) = \sum_{j=1}^{M} 1_{U_{y_j}}(x) g_j(x).$$
If we assume that the sets of root nodes $\{z_k\}$ and $\{y_j\}$ are antichains in X, then the two decompositions differ at most by a reordering of terms.

Proof. Because the first monotonic decomposition refines the second, this means that every z_k is greater than at least one element of $\{y_j\}$ in $(X, \leq)$. In fact, this means there is (not uniquely) an order preserving function $r : \{z_k\} \to \{y_j\}$ such that $r(z_k) \leq z_k$ for every k. On the other hand, the fact that the second monotonic decomposition refines the first means that every y_j is greater than at least one element of $\{z_k\}$

in $(X, \leq)$. Again, there exists an order preserving function $s : \{y_j\} \to \{z_k\}$ such that $s(y_j) \leq y_j$ for every j.

Using this notation,

$$z_m = s(r(z_k)) \leq r(z_k) = y_j \leq z_k,$$

but being an antichain means that $z_m \leq z_k$ implies $m = k$. Thus, $y_j = z_k$ as well. Hence the collection of support sets for the two decompositions coincide up to a permutation of indices. We can therefore compare the associated monotonic functions h_k and g_j on $U_{z_k} = U_{y_j}$. Since both decompositions refine each other, we have that $h_k \leq g_j$ and $h_k \geq g_j$. Antisymmetry for $\leq$ on functions completes the argument. □

The hypotheses for Lemma 6 only yield a sufficient condition. This condition is too restrictive, because we often want to represent dialects that have subset behaviors of larger ones. Such a situation is represented by a monotonic decomposition in which $z_m < z_k$ are both present.

Lemma 7. *Suppose that*

$$f(x) = 1_{U_y}(x)g_1(x) + 1_{U_z}(x)g_2(x) = 1_{U_y}(x)h(x)$$

are two monotonic decompositions of $f : X \to [0, \infty)$ that refine each other, and that $y \leq z$ in X. Then g_2 is identically zero.

Proof. Because the left decomposition refines the right one, we have that $g_1 \leq h$. On the other hand, because the right decomposition refines the left one, we also have that $h \leq g_1$. Thus $g_1 = h$. Since $y \leq z$ in X, this means that $U_z \subseteq U_y$. Therefore,

$$\begin{aligned} f(z) &= 1_{U_y}(z)h(z) \\ &= h(z) \\ &= 1_{U_y}(z)g_1(z) + 1_{U_z}(z)g_2(z) \\ &= g_1(z) + g_2(z) \\ &= h(z) + g_2(z), \end{aligned}$$

whence $g_2(z) = 0$. Since g_2 is assumed to be monotonic, this means that g_2 is identically zero. □

One irritation is that there can be redundancies that complicate minimality.

Definition 4. A monotonic decomposition

$$f(x) = \sum_{k=1}^{N} 1_{U_{y_k}}(x)g_k(x)$$

is called *irredundant* if each of the g_k functions is nonzero for at least one $x \in X$.

Notice that the decomposition in Equation (4) of Example 2 is not irredundant because g_A is identically zero. Simply by excluding any zero terms, every monotonic decomposition refines a unique irredundant monotonic decomposition.

Proposition 4. *The set of irredundant monotonic decompositions of a function $f : X \to [0, \infty)$ from a finite partially ordered set $(X, \leq)$ is a partially ordered set.*

Proof. All that remains after Lemma 4 is antisymmetry. Suppose that

$$f(x) = \sum_{j=1}^{M} 1_{U_{y_j}}(x)g_j(x) = \sum_{k=1}^{N} 1_{U_{z_k}}(x)h_k(x)$$

are two irredundant monotonic decompositions that refine each other. We want to show that these two decompositions are in fact the same, up to reordering of terms.

Let us establish that the sets $\{y_j\}$ and $\{z_k\}$ are identical. To see that the desired result follows from this statement, suppose that $y_j = z_j$ for some j. Then $g_j \leq h_j$ and $h_j \leq g_j$ by the refinement hypotheses, so $g_j = h_j$.

Without loss of generality, suppose that there is a $y \in \{y_j\}$ that is not equal to any z_k. Since the first decomposition refines the second, this means that there must nevertheless be a z_k such that $z_k \leq y$.

Discern two cases: either z_k is equal to an element of y_{j_1} or there is no such element. In the first case, Lemma 7 asserts that $g_{j_1} = 0$ in contradiction to the irredundancy of the first decomposition.

In the second case, although z_k is not equal to any element y_m, nevertheless refinement requires there to be a y_{j_1} such that $y_{j_1} \leq z_k \leq y$. Assuming Lemma 7 does not apply outright to this new situation, we can continue iterating this process to obtain a sequence $y \geq y_{j_1} \geq y_{j_2} \geq \cdots$. Since X is a finite set, this sequence must terminate at some $y' \in \{y_j\}$. Again, because both decompositions refine each other, we must conclude that $y' \in \{z_k\}$. Lemma 7 applies to this situation, and thereby contradicts the irredundancy of at least one of the decompositions. □

Definition 5. (standard, see for instance [17]) Suppose that $(X, \mathcal{T})$ is a topological space for which $\mathcal{T}$ is finite, and that $\mathcal{U} \subseteq \mathcal{T}$ is a cover for X.

An open set $U \in \mathcal{U}$ is called *redundant in $\mathcal{U}$* if there is a subset $\mathcal{V} \subset \mathcal{U}$ such that $U \notin \mathcal{V}$ but $U \subseteq \cup\mathcal{V}$.

A cover with no redundant open sets is called an *irredundant cover*.

Efficient algorithms for finding irredundant covers have been known for a long time [17]. Irredundant monotonic decompositions correspond to dialects with nonzero probabilities, and so are useful in helping to identify candidate dialect decompositions. It is informative to know the number of dialects that could be present in a given dataset, which Corollary 2 below relates to minimal monotonic decompositions. As a practical matter, irredundant monotonic decompositions are especially useful because even though they are not unique, they are unambiguous about the *number* of dialects involved.

Corollary 2. *Because each dialect decomposition corresponds to a monotonic decomposition according to Proposition 2, Theorem 1 implies that a lower bound on the number of dialects given for a probability distribution as expressed by Equation* (2) *is the minimum number of terms in a minimally refined monotonic decomposition.*

Theorem 2 is an immediate consequence of Lemmas 8 and 9, which follow.

Lemma 8. *All minimal irredundant monotonic decompositions of a given function have the same support sets and hence have the same number of terms.*

This is different from the related situation of finding minimal irredundant decompositions of logic functions. Logic functions are known that have minimal irredundant decompositions into sums with different numbers of terms [18].

Proof. Suppose that

$$f(x) = \sum_{y \in R \subseteq X} 1_{U_y}(x) g_y(x)$$

is a minimal irredundant monotonic decomposition of an arbitrary function f. The statement to be proven is that the R set in the equation is the same for all minimal irredundant monotonic decompositions of f. More explicitly, if $y \in R$, so that $1_{U_y}(x)g_y(x)$ is a term in the minimal irredundant monotonic decomposition above, then any other irredundant monotonic decomposition must also have a term of the form $1_{U_y}(x)h_y(x)$.

Suppose that $y \in X$ is such that $f(x) = 0$ for all $x \leq y$. Then every monotonic decomposition (irredundant or not) of f must contain a term of the form $1_{U_y}(x)h_y(x)$.

Suppose that $y \in X$ is such that there is an $x \in X$ such $x \leq y$ and $f(x) \neq 0$. We can rewrite the monotonic decomposition as

$$f(x) = \sum_{v \in R: v < y} 1_{U_v}(x) g_v(x) + \sum_{w \in R: y \leq w} 1_{U_w}(x) g_w(x) + \sum_{z \in R: z \not\leq y, y \not\leq z} 1_{U_z}(x) g_z(x).$$

Because $g_y(y) \neq 0$ by irredundancy, the middle term must be positive. This means that the above decomposition leads to the inequality

$$f(y) > \sum_{v \in R: v < y} 1_{U_v}(y) g_v(y).$$

We can take this inequality a bit further. The Proposition follows if there are is a subset R' of those $v \in R$ satisfying both $v < y$ and

$$f(y) > \sum_{v \in R: v < y} 1_{U_v}(y) g_v(y) = \sum_{u \in R'} f(u). \tag{6}$$

This claim can be proven by contradiction; assume that there is a subset $R' \subseteq \{v \in R : v < y\}$ such that

$$f(y) = \sum_{u \in R'} f(u).$$

If this is the case, using the fact that the g_v functions sum to $f(u)$ on each $u \in R'$, we can choose the g_v functions to take the same value at y without violating monotonicity. Thus g_y has to be zero because the sum of all the $g_v(y)$ is equal to $f(y)$, a contradiction with irredundancy. Obviously a smaller $f(y)$ forces $g_y(y) = 0$ as well. In any case, this also contradicts irredundancy. Thus $f(y)$ is strictly greater than that, as Equation (6) claims.

Notice that the last sum in Equation (6) does not depend on g_v. A term involving U_y in an irredundant monotonic decomposition is therefore determined directly by the values of f. Therefore, any other monotonic decomposition will be subject to the same situation and therefore will need to contain a term involving y. □

Lemma 9. *Suppose that $(X, \mathcal{T})$ is a topological space for which $\mathcal{T}$ is finite, that $\mathcal{U}$ is a cover which refines an irredundant cover $\mathcal{V}$. Then $\#\mathcal{V} \leq \#\mathcal{U}$.*

Proof. Let $V \in \mathcal{V}$. The hypotheses imply there is a $U \in \mathcal{U}$ such that $U \subseteq V$. Let us establish this claim by contradiction. Suppose that no $U \in \mathcal{U}$ is a subset of V. Because $\mathcal{U}$ is a cover, there is a collection $\mathcal{U}' \subseteq \mathcal{U}$ such that $V \subseteq \cup\mathcal{U}'$. Since $\mathcal{U}$ refines $\mathcal{V}$, each $U' \in \mathcal{U}'$ is a subset of some $V' \in \mathcal{V}$. Consider the subset

$$\mathcal{V}' := \{V' \in \mathcal{V} : U' \subseteq V' \text{ for some } U' \in \mathcal{U}'\} \subseteq \mathcal{V}.$$

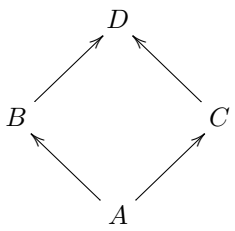
Evidently $V \subseteq \cup\mathcal{V}'$. Recalling that we assumed no $U \in \mathcal{U}$ is a subset of V, we must conclude that $V \neq \cup\mathcal{V}'$, which contradicts the irredundancy of $\mathcal{V}$.

We complete the argument by induction on $\#\mathcal{V}$.

- Base case: Suppose that $\#\mathcal{V} = 1$. Because both $\mathcal{V}$ and $\mathcal{U}$ both cover X, and $\mathcal{V}$ is evidently nonempty, then $\mathcal{U}$ must also be nonempty.
- Induction case: Suppose that the Lemma has been established for all $\mathcal{V}$ with $\#\mathcal{V} \leq n$ for some integer n. Suppose that $\mathcal{V}$ is an irredundant cover containing $n+1$ elements, $V_0, \ldots, V_n$. Consider the subspace of $(X, \mathcal{T})$ covered by $V_0, \ldots, V_{n-1}$. By the claim proven above, there is a subset $\mathcal{U}' \subseteq \mathcal{U}$ that both covers $V_0 \cup \cdots \cup V_{n-1}$ and refines the cover $\{V_0, \ldots, V_{n-1}\}$. The induction hypothesis applied to this situation asserts that $\#\mathcal{U}' \geq n$. By the irredundancy of $\mathcal{V}$, we must have that $V \not\subseteq (V_0 \cup \cdots \cup V_{n-1})$. Because each element of $\mathcal{U}'$ is a subset of at least one of the $V_0, \ldots, V_{n-1}$, we have that $V \not\subseteq \cup\mathcal{U}'$ as well. Therefore, to be a cover of X, $\mathcal{U}$ must have at least one more element than $\mathcal{U}'$. Hence,

$$\#\mathcal{U} \geq \#\mathcal{U}' + 1 \geq n + 1 = \#\mathcal{V}.$$ □

Example 3. Suppose that $X = \{A, B, C, D\}$ is the partial order defined by the Hasse diagram



According to Theorem 2, all minimally refined irredundant monotonic decompositions of an arbitrary function $f : X \to [0, \infty)$ have the same number of terms. We can demonstrate

this fact by reasoning about monotonic decompositions directly.

If $f(A) \neq 0$, then every irredundant monotonic decomposition of f must contain a term of the form $1_{U_A} g_A$ where $g_A(A) = f(A)$. Therefore, without loss of generality, we may assume that $f(A) = 0$. Also, without loss of generality, we may assume that $f(B) \leq f(C)$.

If $f(B) = 0$, then there are no choices to be made in the decomposition of f:

- If $f(C) = 0$, the decomposition has at most one term of the form $1_{U_D} f(D)$.
- If $f(D) > f(C) > 0$ the decomposition contains a term of the form $1_{U_D}(f(D) - f(C))$.
- Otherwise the decomposition has only one term.

If instead $f(B) \neq 0$, whether we start the decomposition using B or C does not change the resulting number of terms in the decomposition. This happens because after removing the contribution from a term supported on B or C results in a new function that decomposes as above. Notice that this latter situation is exactly what happened in Example 2.

While Theorem 2 handles the case of minimally refined irredundant monotonic decompositions, which have useful implications for determining the number of dialects, *maximally* refined irredundant monotonic decompositions also exist.

Proposition 5. *There is a unique maximally refined irredundant monotonic decomposition of a function $f : X \to \mathbb{Z}^+$ from a finite partially ordered set $(X, \leq)$ to the nonnegative integers, namely*

$$f(x) = \sum_{y \in X} \sum_{i=1}^{f(y)} 1_{U_y}(x) 1_{\{y\}}(x). \tag{7}$$

The maximally refined monotonic decomposition is rather uninformative, because it means that each dialect contains exactly one file.

Proof. Because X is assumed to be finite and each of the monotonic functions in a monotonic decomposition produce nonnegative integers, the set of monotonic decompositions is finite. Therefore, Lemma 4 implies that there are maximal and minimal monotonic decompositions under refinement.

Suppose that we have an arbitrary monotonic decomposition of f,

$$f(x) = \sum_{k=1}^{N} 1_{U_{y_k}}(x) g_k(x). \tag{8}$$

We must show that Equation (7) refines this decomposition.

Close inspection of the sum in Equation (7) reveals that it only contains terms involving U_y if $f(y) > 0$. While the outer sum would seem to imply that terms involving U_y will be present for all $y \in X$, the inner sum prevents the inclusion of any term for which $f(y) = 0$. Hence, Equation (7) defines an irredundant monotonic decomposition.

Given this observation, consider a $y \in X$ for which $f(y) > 0$. Necessarily, there must be a term in Equation (8) for which $y \in U_{y_k}$ and $g_k(y) > 0$. Therefore, $U_y \subseteq U_{y_k}$. Moreover, since each of the monotonic functions in Equation (7) simply take the value 1 on exactly one element of X, we have that $1_{\{y\}} \leq g_k(y)$. □

Remark 1. If we instead permit redundancies in Proposition 5, then we may add terms with the zero function arbitrarily. While these monotonic decompositions all refine each other, this precludes uniqueness of such a decomposition.

Remark 2. If we instead consider $f : X \to [0, \infty)$ in Proposition 5, then the inner sum in Equation (7) becomes infinite. There is no maximally refined monotonic decomposition in this case.