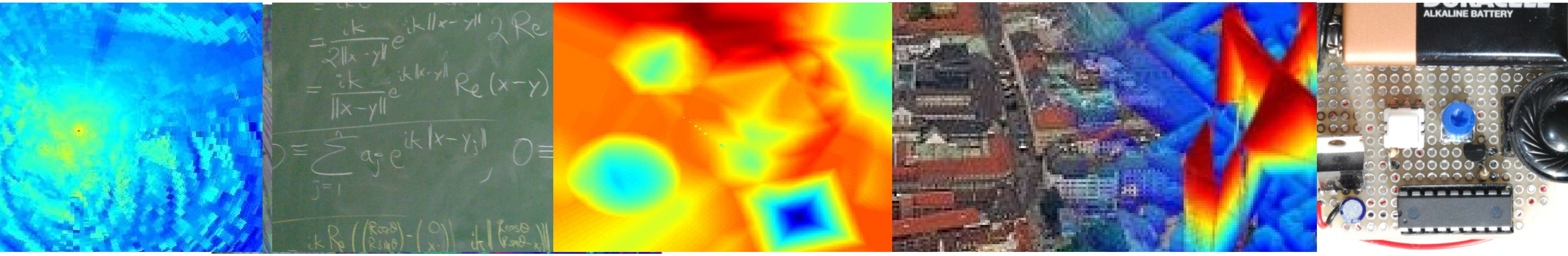
Unsupervised Clustering of File Dialects *according to* Monotonic Decompositions of Mixtures



Michael Robinson, Tate Altman,
Denley Lam, Letitia Li

DISTRIBUTION STATEMENT A. Approved for Public Release

This material is based upon work supported by the Defense Advanced Research Projects Agency (DARPA) under Contract No. HR001119C0072. Any opinions, findings and conclusions, or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of DARPA.

Acknowledgments

- Additional collaborators:
  - Cory Anderson (BAE Systems)
  - Kris Ambrose, Steve Huntsman, Allyson O’Brien, Matvey Yutin (all formerly at BAE Systems)
- Students:
  - Ken Ewing, Natalie Tsuei (American Univ.)
- Data from:
  - G. J. J. van den Burg (Alan Turing Inst.)
  - Tim Allen (NASA/JPL)
  - Peter Wyatt (PDF Association)
- Funding:
  - Sergey Bratus (DARPA/I2O) SafeDocs

Problem definition

What do we mean when we say that an *ad hoc* file format has multiple dialects?

We want to find these dialects in a way that is both **accurate** and easily **explainable**

- **Accurate** models fit the data
- Explainable models are **parsimonious**
- They contain the minimum number of dialects necessary to characterize the observed behavior.

Measuring software behavior

- Our methodology does not look "into" the contents of a file… parsers already exist that can do that!
- Parsers produce measurable output when they consume a file, a *message*
  - It's usually more than simply "parsed OK" or "error"
  - We don't need to consider, say, rendered output
  - Categorical data suffices; "C `enum` type"
- File behavior consists of a set of triples:
  - Parser ID
  - File ID
  - Message

File behaviors ascendant

- The methodology in this talk has been successfully applied to
  - **CSV files**
  - **NITF files**
  - **PDF files**

    This talk
  - MPEG files
  - JPEG files
  - English text files
- We are currently working to apply it further to
  - Ontology tagged error matrices (generally)
  - Byte histograms from file contents and traces
- Takeaway: This is a general tool with substantial practical power!

What is a message?

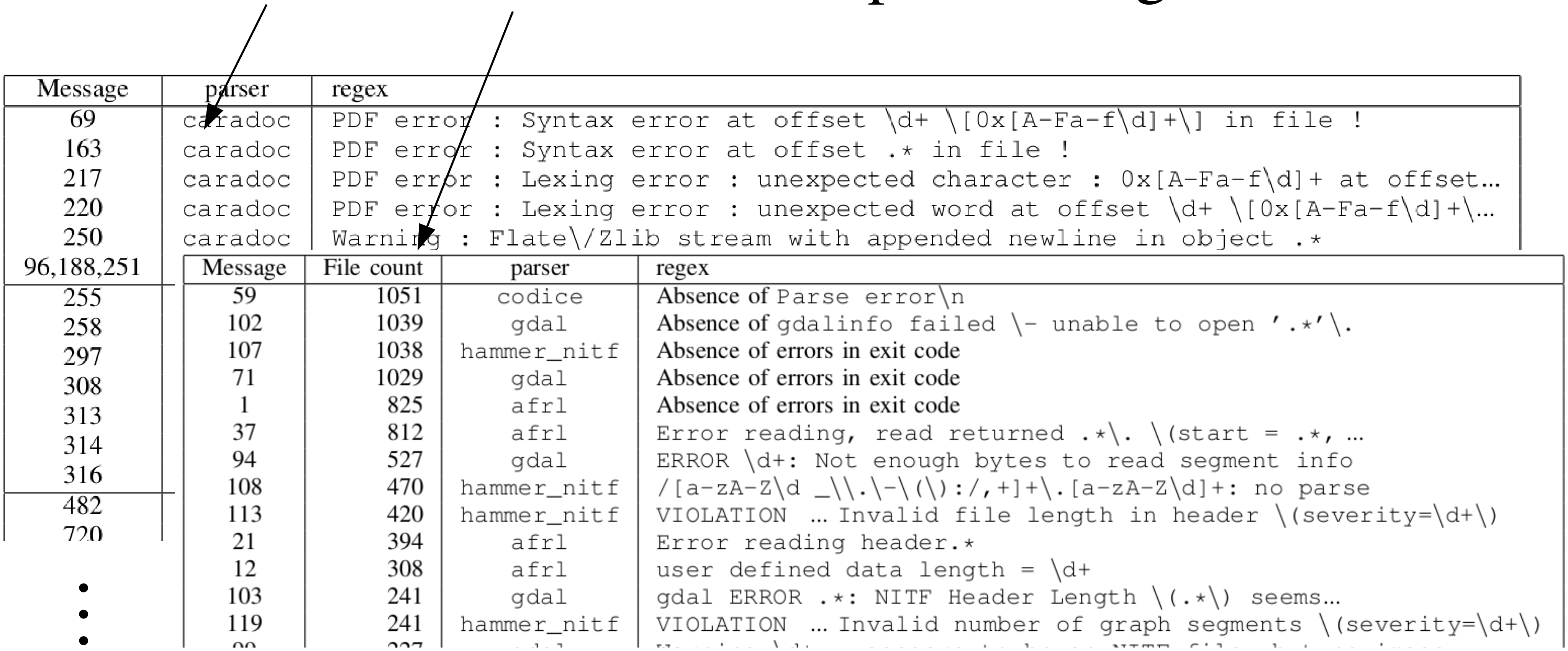
- For CSV:
  - Which encoding? ASCII, UTF-8, etc.
  - Which delimiters (if any)? Comma, space, etc.
  - Which kind of quotes (if any)?

What is a message?

- For CSV: character regexes and simple lexers

What is a message?

- For CSV: character regexes and simple lexers
- For PDF or NITF, more complicated regexes



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		
255		
258		
297		
308		
313		
314		
316		
482		
720		

⋮

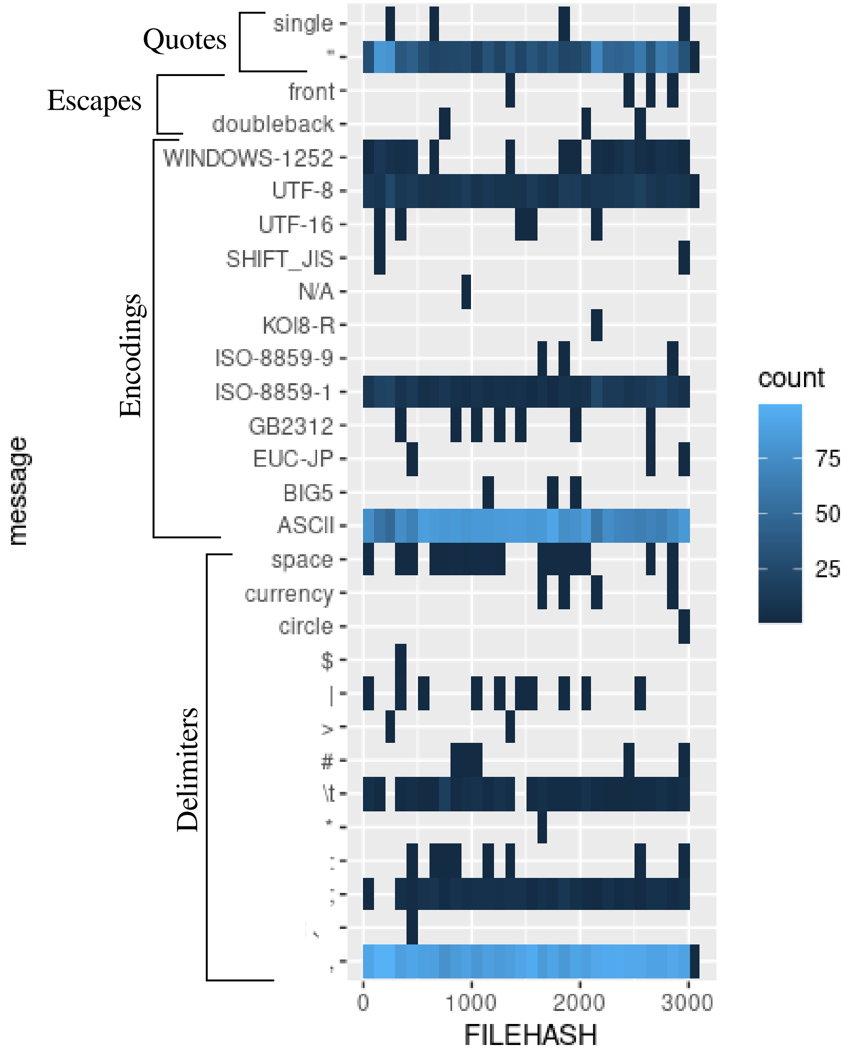
Message	File count	parser	regex
59	1051	codice	Absence of Parse error\n
102	1039	gdal	Absence of gdalinfo failed \- unable to open ′.⋆′\.
107	1038	hammer_nitf	Absence of errors in exit code
71	1029	gdal	Absence of errors in exit code
1	825	afrl	Absence of errors in exit code
37	812	afrl	Error reading, 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 \(severity=\d+\)

⋮

CSV example data

- Data: CSV files culled from the wild
  https://github.com/alan-turing-institute/CSV_Wrangling
- Messages obtained from CleverCSV:
  https://github.com/alan-turing-institute/CleverCSV
  - 14 Delimiters
  - 3 Quote Characters
  - 3 Escape Characters
  - 13 distinct text encodings

Joint behaviors: message *patterns*



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`

Message patterns are **explainable**

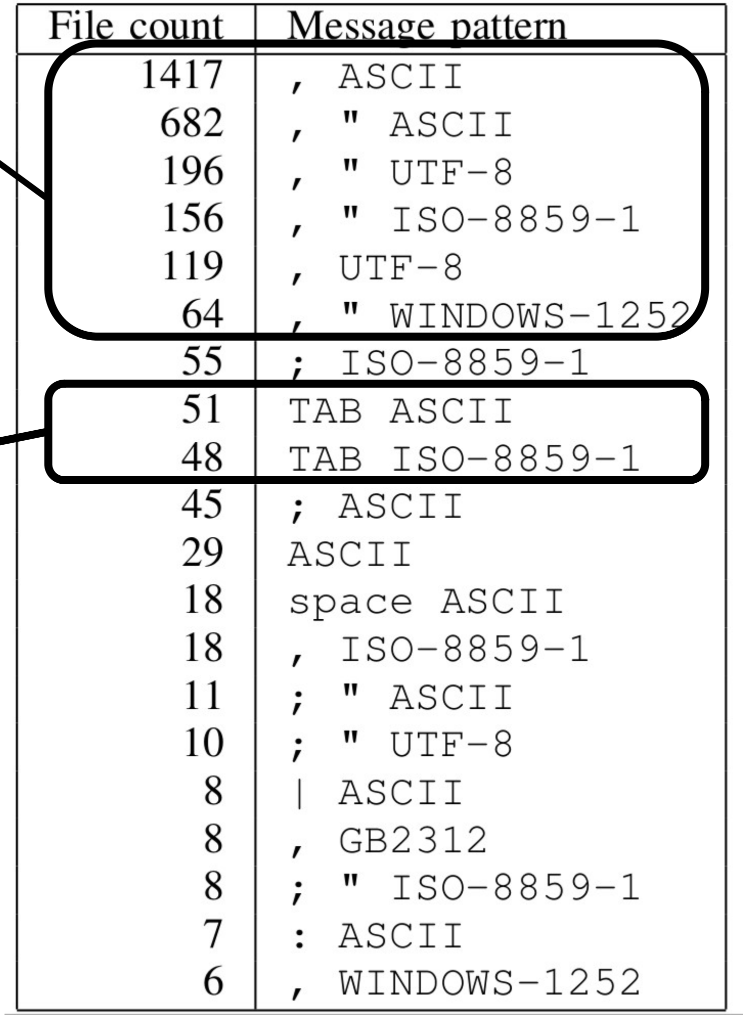
- Most CSV files are as you might expect:

  comma **s**eparated **v**alues

- However many are not...

wat? This must be Excel's fault…

… so perhaps there are several *dialects* of CSV files present



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

Message pattern probability (Take 1)

What's the probability that a file from dialect *A* exhibits a set of messages *K*?

This is easy if we assume* messages are independent when conditioned on dialect:

$$P(K|A) = p_0^{\#(K\cap M_A^c)}(1-p_0)^{\#(K^c\cap M_A^c)} \times \longleftarrow \text{background less frequent messages}$$

$$p_A^{\#(K\cap M_A)}(1-p_A)^{\#(K^c\cap M_A)}. \longleftarrow \text{dialect } A \text{ more frequent messages}$$

Message didn't happen

Message did happen

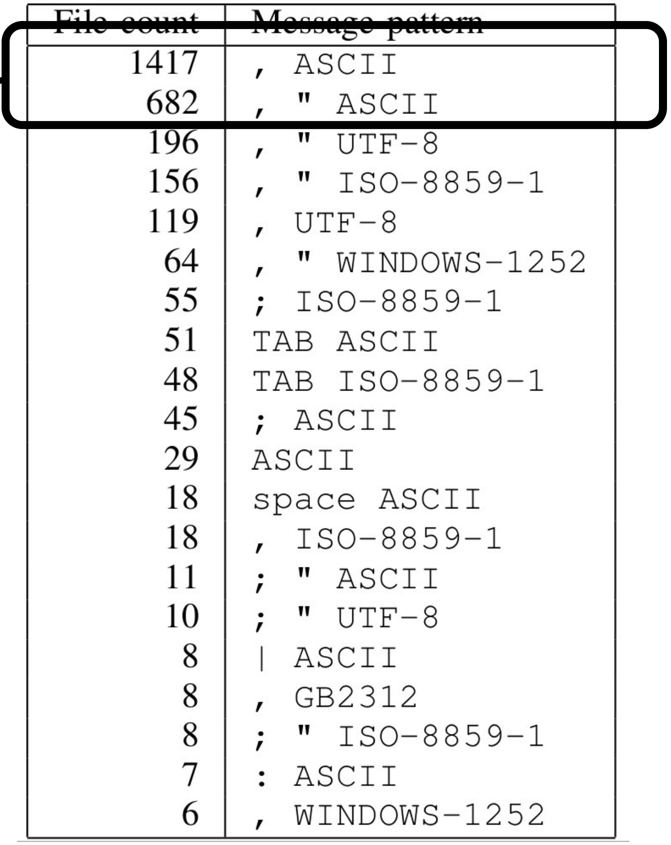
*Hold that thought!
Challenging one’s assumptions is **important**!

Message patterns are ordered by subset

- Theorem: Under our theoretical model, patterns with **more** messages are **less** frequent
- Monotonicity means that subset ordering is exactly opposite file count ordering

{, ASCII} ⊆ {, “ ASCII}
Counts: 1417 > 682

“Message counts vary occur because files vary randomly”



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

Message patterns are ordered by subset

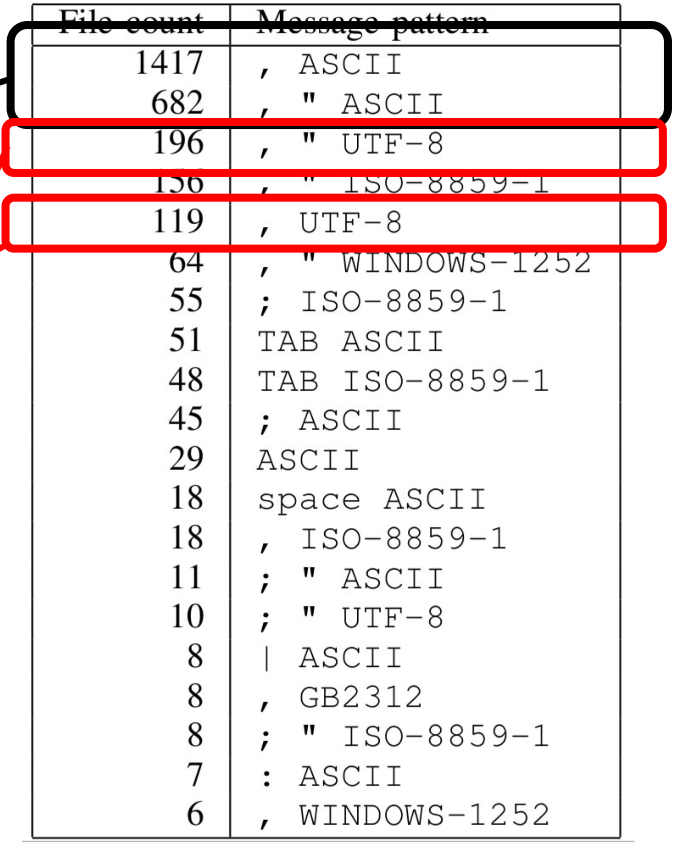
- Theorem: Under our theoretical model, patterns with **more** messages are **less** frequent
- Monotonicity means that subset ordering is exactly opposite file count ordering

{, ASCII} ⊆ {, “ ASCII}
Counts: 1417 > 682

- Monotonicity doesn’t always happen… for good* reason!

{, UTF−8} ⊆ {, “ UTF−8}
Counts: 119 < 196

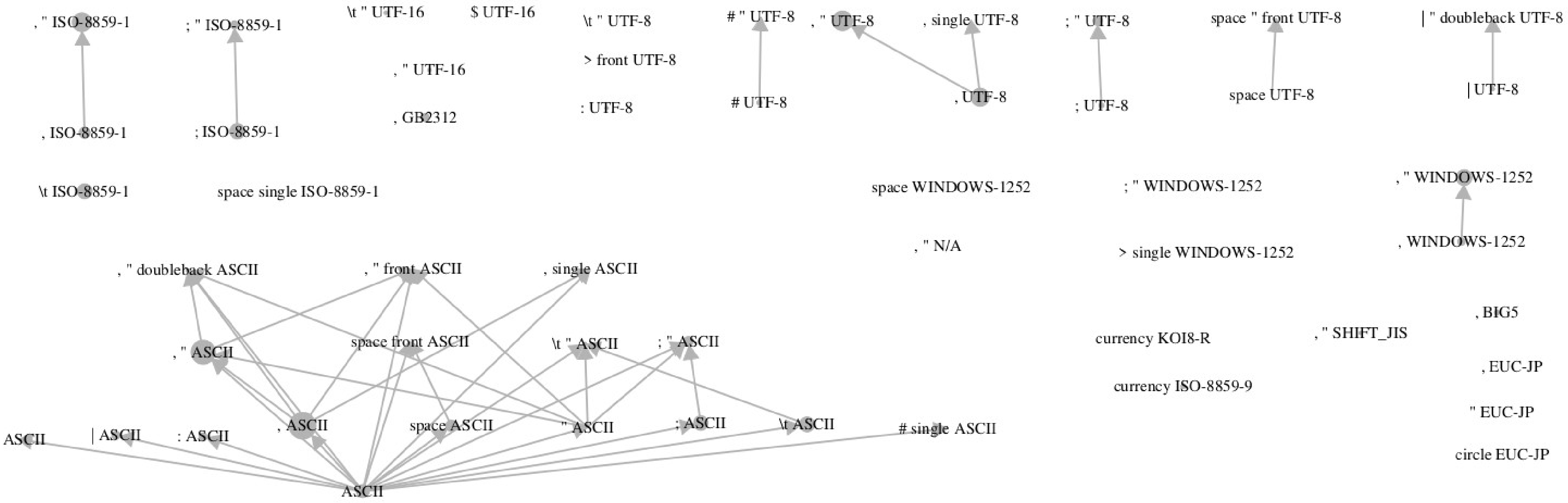
*Files are **not** random, they have structure



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

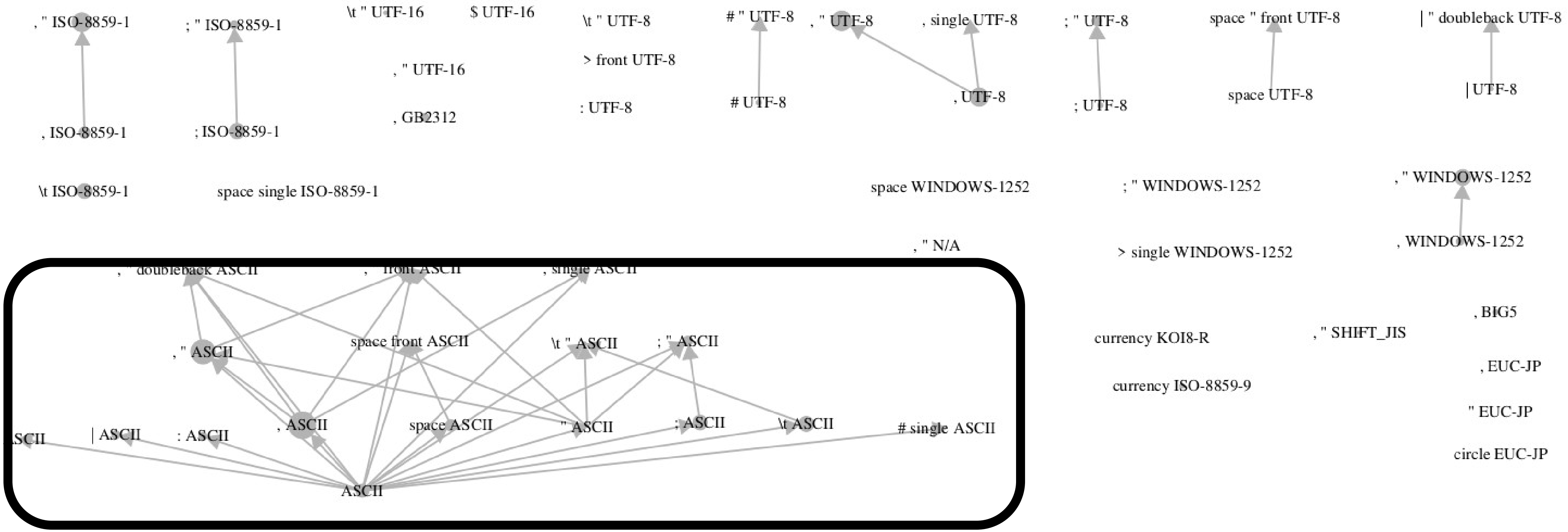
CSV message pattern partial order

- Vertices = distinct message patterns
- Vertices sized by file count
- Edges directed according to message pattern order



CSV message pattern partial order

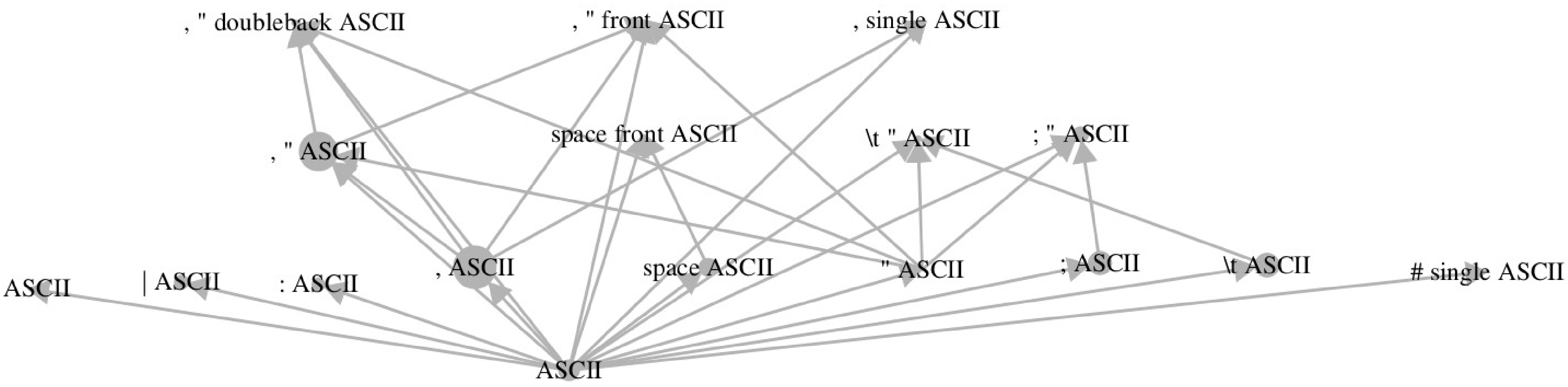
- Vertices = distinct message patterns
- Vertices sized by file count
- Edges directed according to message pattern order



ASCII files

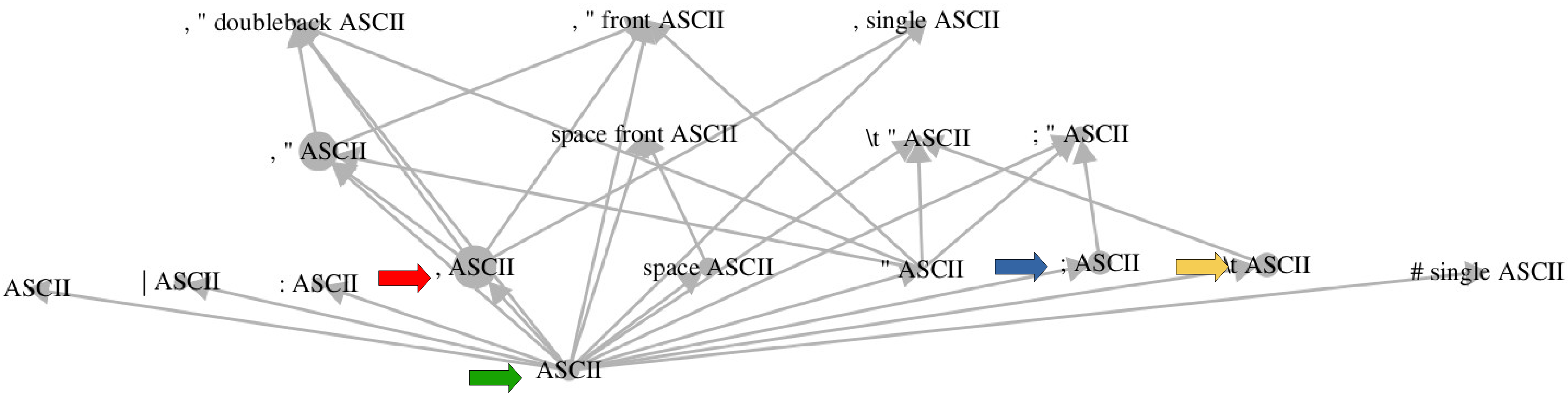
ASCII dialect of CSV

- Vertices = distinct message patterns
- Vertices sized by file count
- Edges directed according to message pattern order



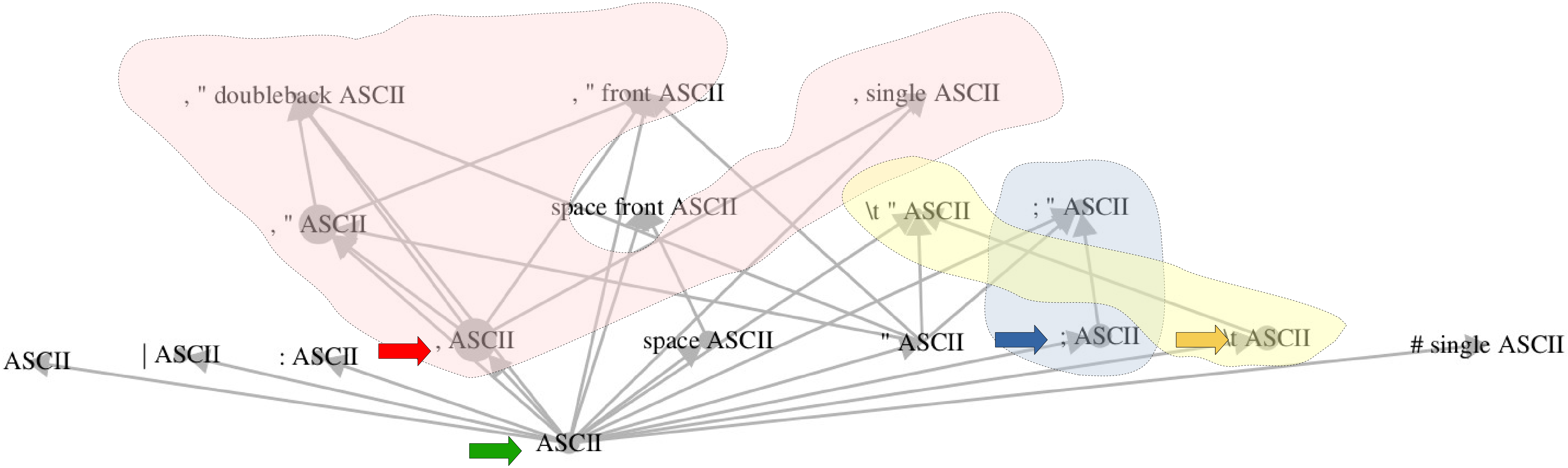
ASCII dialect of CSV

- Dialects appear to correspond to places where the file count is not monotonic
  - Violations to monotonicity are marked
  - Each message pattern in question defines a dialect



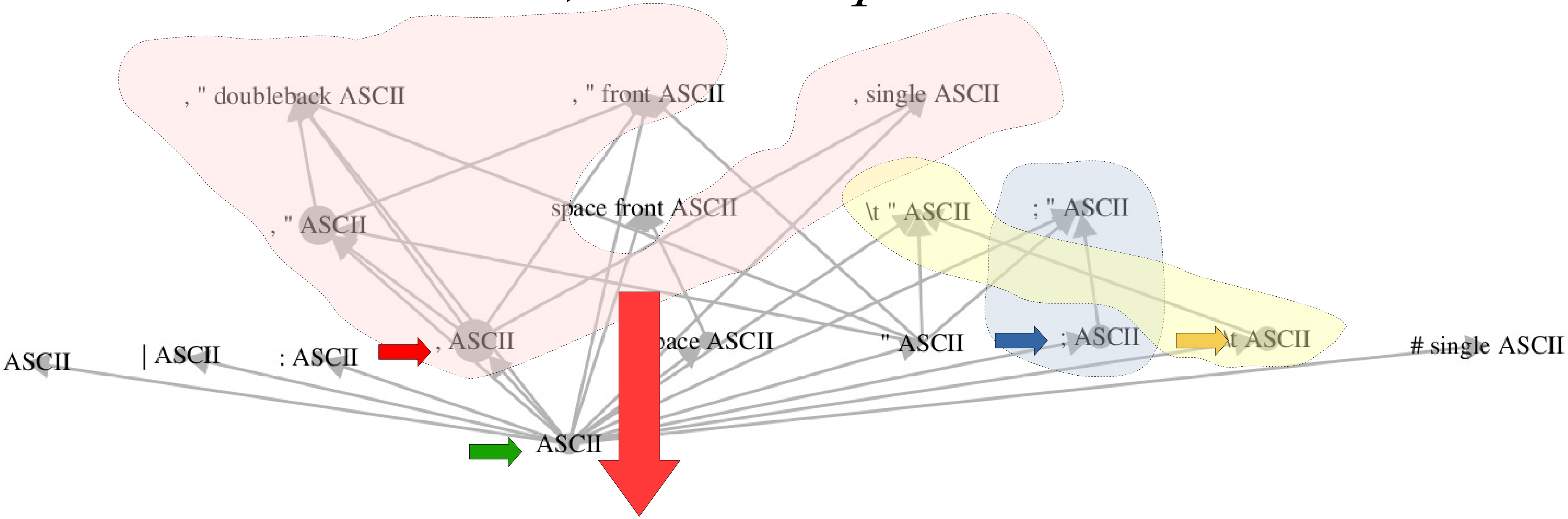
ASCII dialect of CSV

- Message patterns “containing” a monotonicity violation could be part of that dialect



Required messages

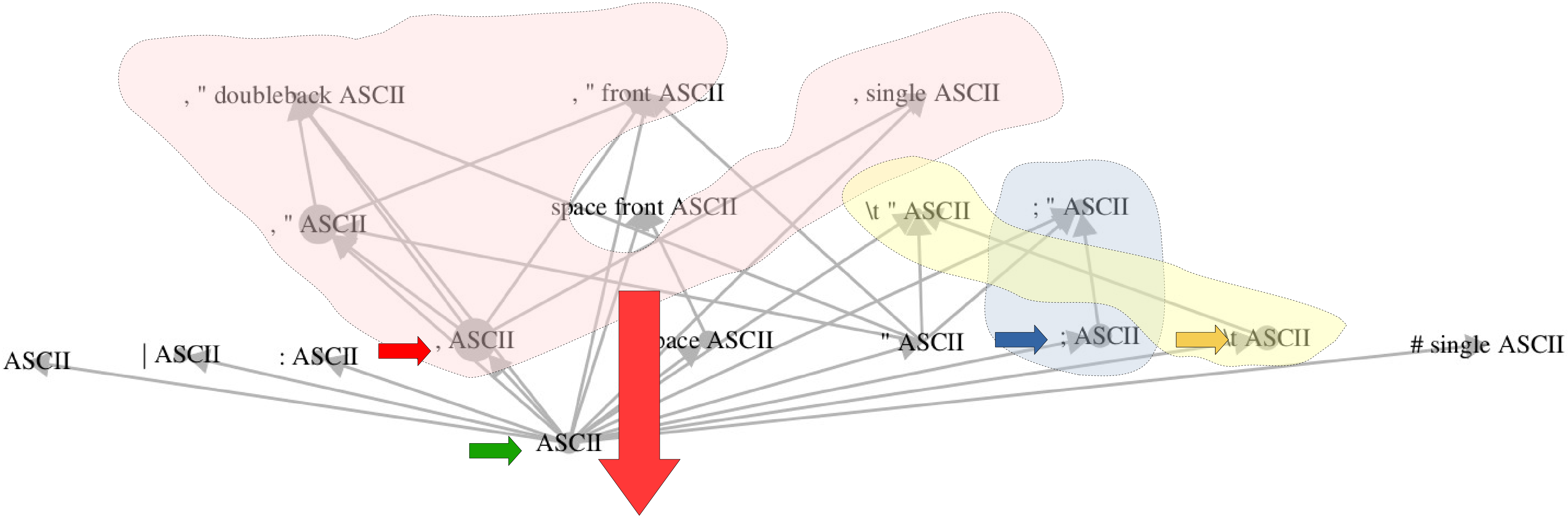
- Message patterns “containing” a monotonicity violation could be part of that dialect
- The minimal set of messages in each dialect characterize it, and are *required* for that dialect



Dialect consists of some files producing at least `, ASCII`

Message pattern ambiguity

- Caution: message patterns **do not guarantee** that a file is of a given dialect
  - Non-tabular text can produce `, ASCII` without being a CSV file



Dialect consists of **some files** producing at least `, ASCII`

Message pattern probability (Take 2)

A corpus with many dialects using an *independent mixture model*

Frequency of dialect *A* in dataset

$$P(k_1, k_2, \ldots, k_n) = \sum_A P(k_1, k_2, \ldots, k_n | A) P(A)$$

message pattern probability in dialect *A*

each dialect is a term in this sum

The same message pattern can appear in multiple dialects, though with probability < 1 in each case

Message pattern probability (Take 2)

A corpus with many dialects using an *independent mixture model*

Frequency of dialect *A* in dataset

$$P(k_1, k_2, \ldots, k_n) = \sum_A P(k_1, k_2, \ldots, k_n | A) P(A)$$

message pattern probability in dialect *A*

Insight: Messages in each dialect are independent once a set of *dialect required messages* occur first

Required messages must occur!

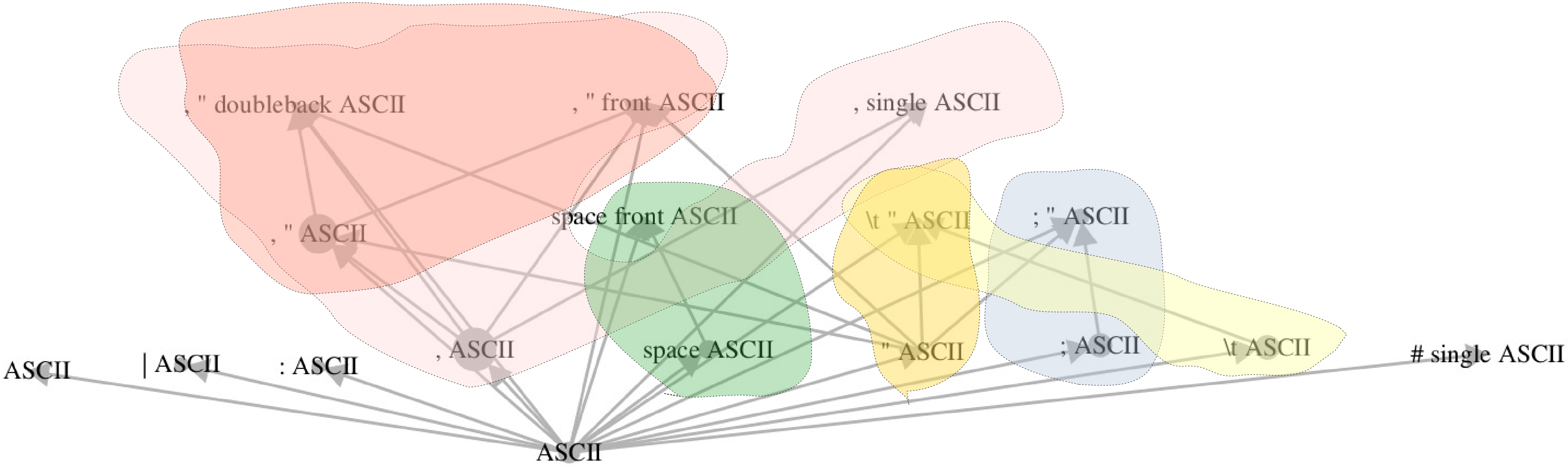
$$P(k_1, k_2, \ldots, k_n | A) = \begin{cases} 0 \text{ if } k_j = 0 \text{ and } K_j \in R_A, \\ P(k_1|A) \cdots P(k_i|A) P(k_{i+1} = 1, \ldots | A) \\ \text{otherwise.} \end{cases}$$

Non-required messages are independent

Ambiguity is present and useful

Many dialect decompositions may be consistent with the observed data : **accuracy** is required

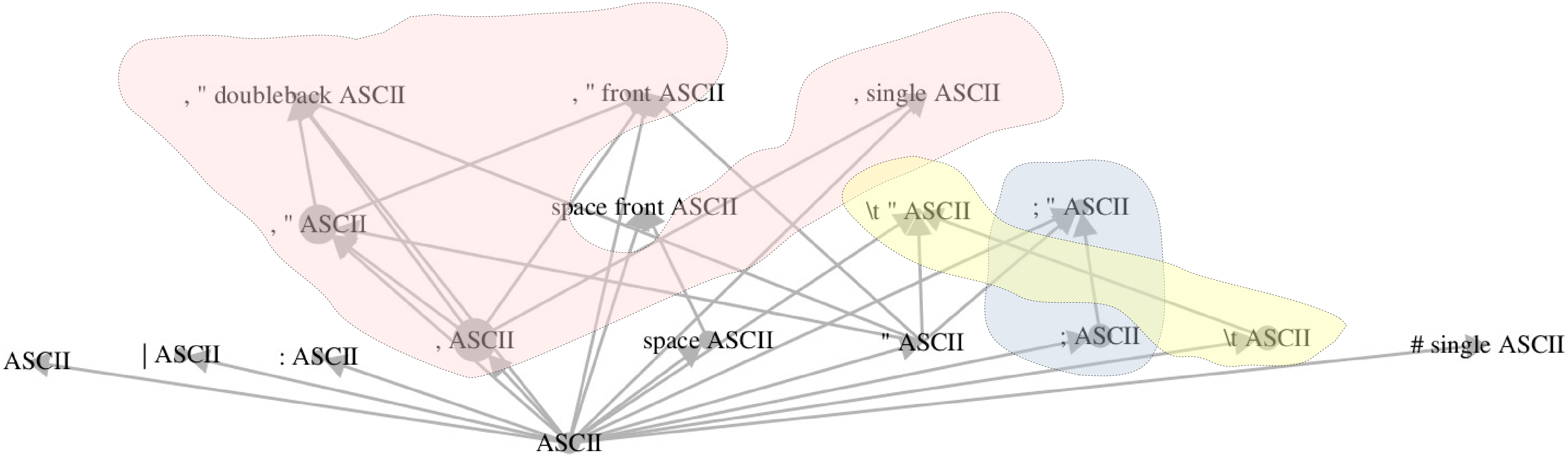
Some decompositions have many dialects...



Ambiguity is present and useful

Many dialect decompositions may be consistent with the observed data : **accuracy** is required

… while others have fewer



The one shown here is the coarsest one that is consistent with the data – the most **explainable**

Coarsest dialect decomposition exists

Many dialect decompositions may be consistent with the observed data

Candidate message pattern probabilities

$$\begin{aligned} P(k_1, k_2, \ldots, k_n) &= \sum_A P(k_1, k_2, \ldots, k_n | A) P(A) \\ &= \sum_A 1_{U_{R_A}}(k_1, \ldots, k_n) g_A(k_1, \ldots, k_n) \end{aligned}$$

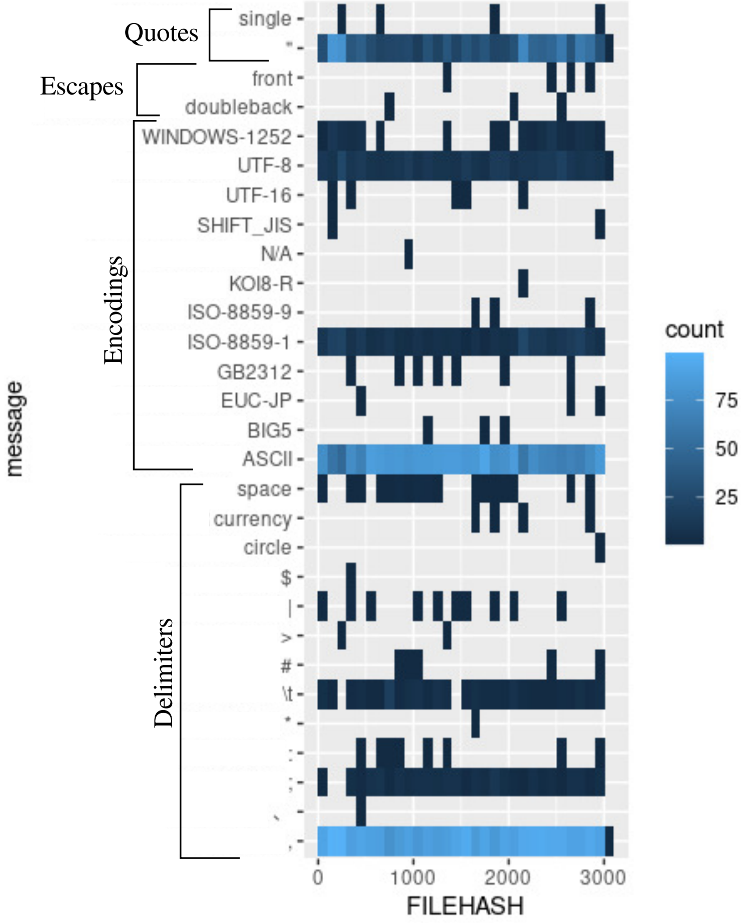
Candidate required message sets, found greedily

<u>Theorem</u>: There is a unique, coarsest decomposition into candidate dialects; this can be found **algorithmically**

<u>Theorem</u>: The actual dialects are bounded below by (and are strictly finer than) the candidate dialects

CSV candidate dialects

The coarsest decomposition yields dialects that are exactly what one would expect… explanation is easy



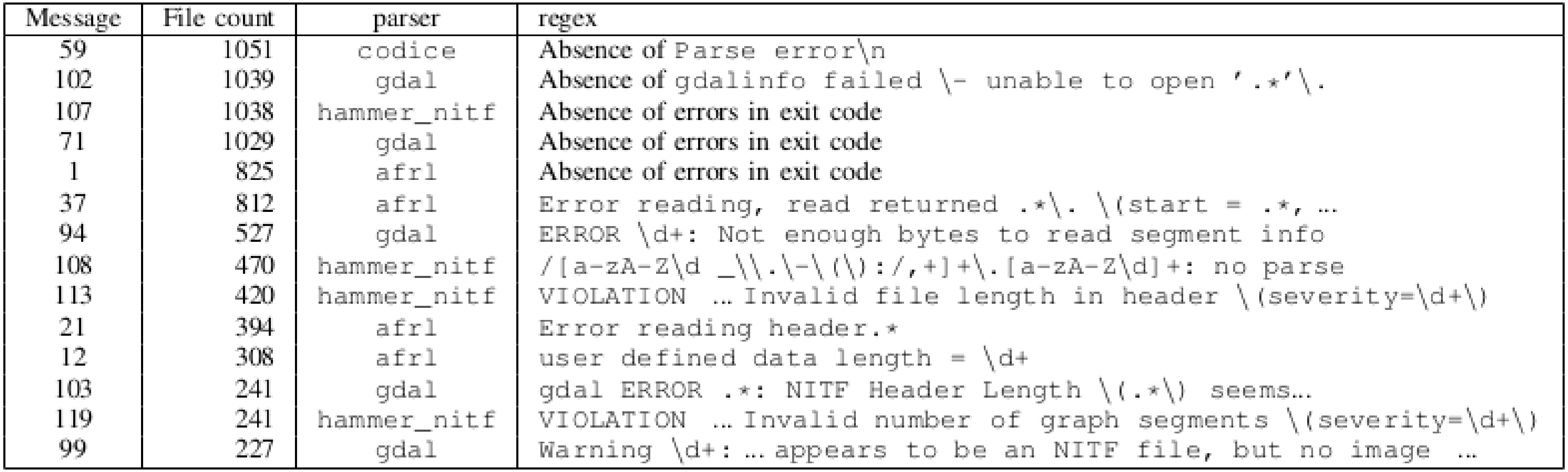
File count at root	Required messages
1388	, ASCII
119	, UTF-8
77	, " UTF-8
18	, ISO-8859-1
7	; " UTF-8
22	TAB ASCII
138	, " ISO-8859-1
48	TAB ISO-8859-1
58	, " WINDOWS-1252
29	ASCII
55	; ISO-8859-1
16	; ASCII
8	, GB2312
6	, WINDOWS-1252

English text delimited w/ commas

The rest are English text delimited w/ other chars

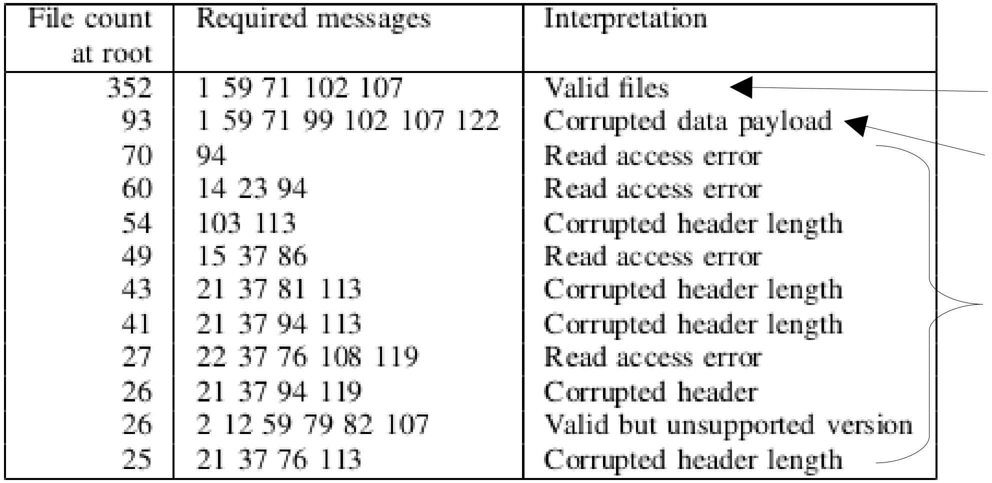
Michael Robinson

Encore: NITF candidate dialects



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 …

Candidate dialects capture human-interpretable file behaviors



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

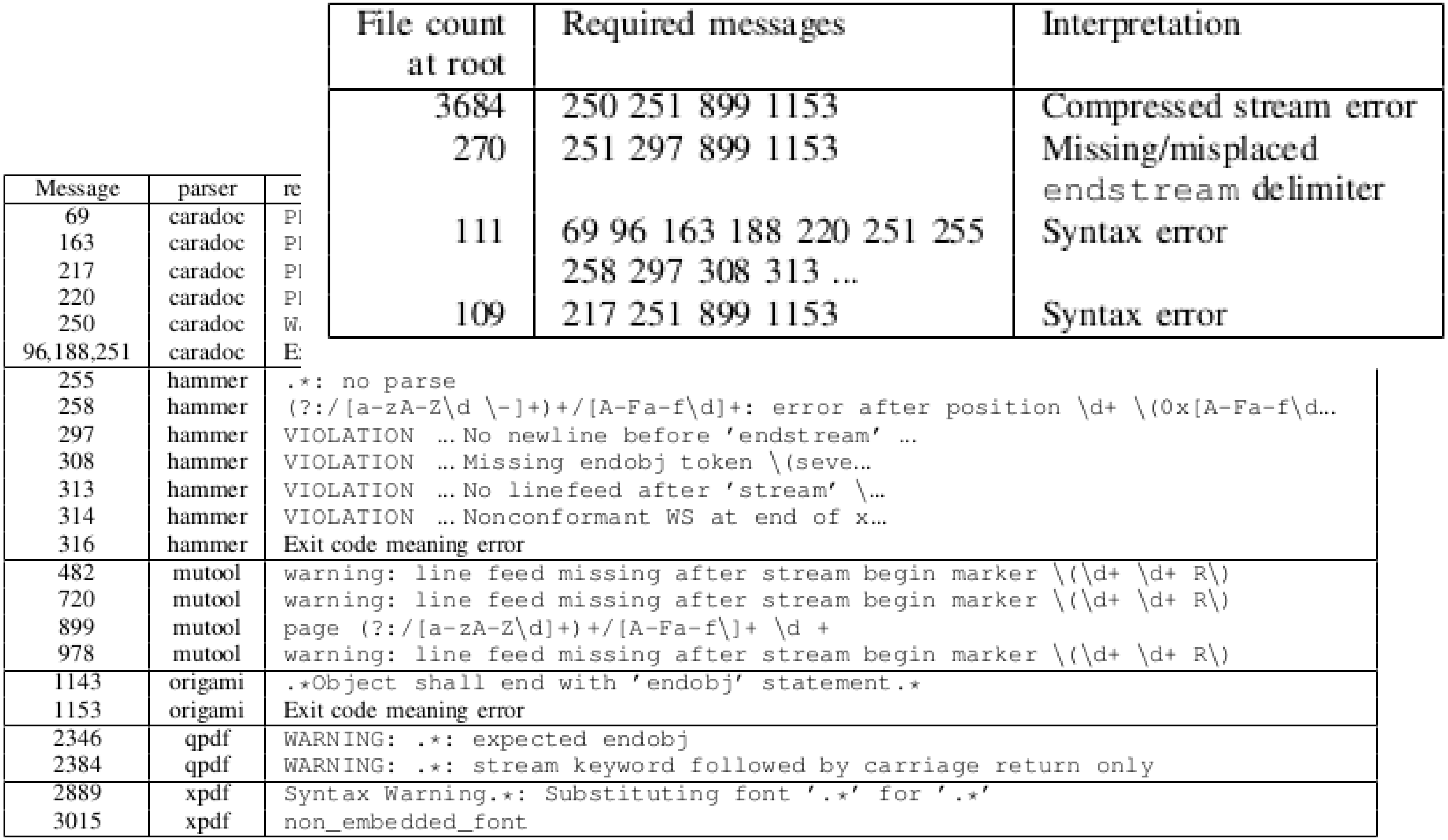
Valid files

Valid but unreadable

Invalid

Encore^2: PDF candidate dialects

Again, candidate dialects identify human-interpretable behaviors



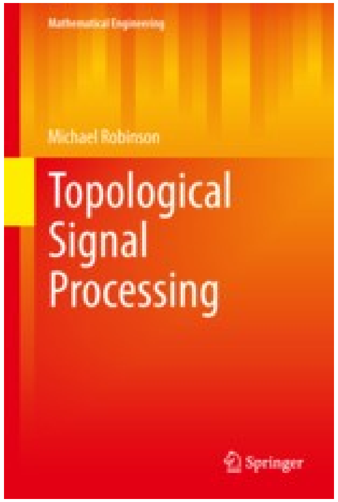
File count at root	Required messages	Interpretation
3684	250 251 899 1153	Compressed stream error
270	251 297 899 1153	Missing/misplaced `endstream` delimiter
111	69 96 163 188 220 251 255 258 297 308 313 ...	Syntax error
109	217 251 899 1153	Syntax error

Message	parser	re
69	caradoc	PI
163	caradoc	PI
217	caradoc	PI
220	caradoc	PI
250	caradoc	W.
96,188,251	caradoc	E:
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`

Conclusions

- File behavior can be characterized by collecting parser responses through their output messages
- Files of a dialect exhibit similar behaviors that can be identified by probabilistic clustering
- What a dialect means is ambiguous, but the mathematics supports this ambiguity
  - There are many **accurate** dialect decompositions
  - There is a well-defined, unique coarsest decomposition
  - The coarsest dialect decomposition is **easily explainable**
- This methodology can be easily retooled to handle many different file formats

To learn more...


Mathematical Engineering

Michael Robinson

Topological Signal Processing

Springer

Michael Robinson

michaelr@american.edu

http://drmichaelrobinson.net

Relevant references:

doi:10.1109/SPW53761.2021.00032

doi:10.1109/SPW54247.2022.9833862

arXiv:2105.01690

Software:

https://github.com/kb1dds

https://www.youtube.com/watch?v=i3wl2jdIZv8