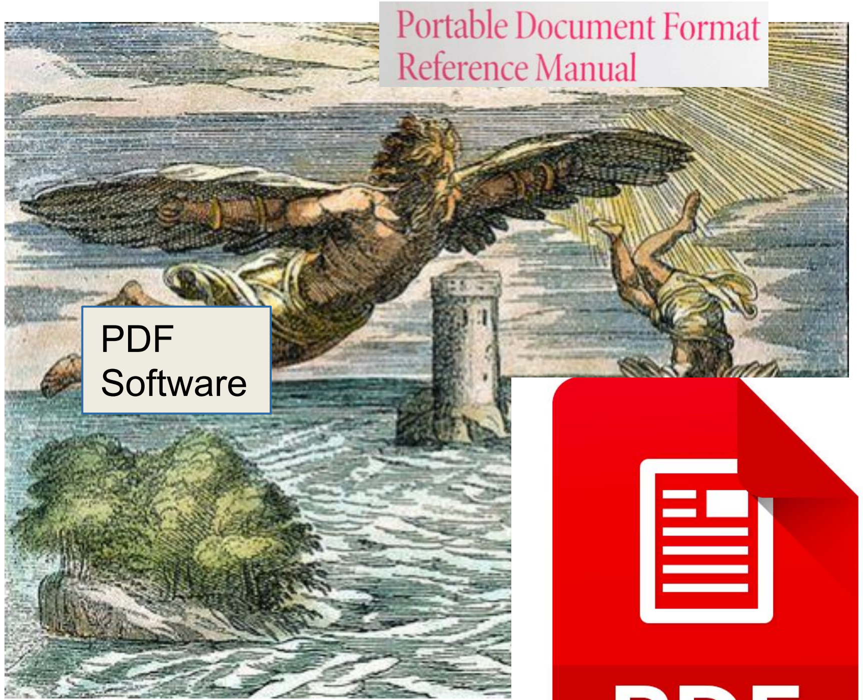
Research Report: ICARUS: Understanding De Facto Formats By Way of Feathers and Wax

Sam Cowger, Yerim Lee, Nichole Schimanski, Mark Tullsen, **Walt Woods**, Richard Jones, EW Davis, William Harris, Trent Brunson, Carson Harmon, Bradford Larsen, Evan Sultanik

De Facto Corpuses

- Widely adopted formats can expand beyond original specifications.

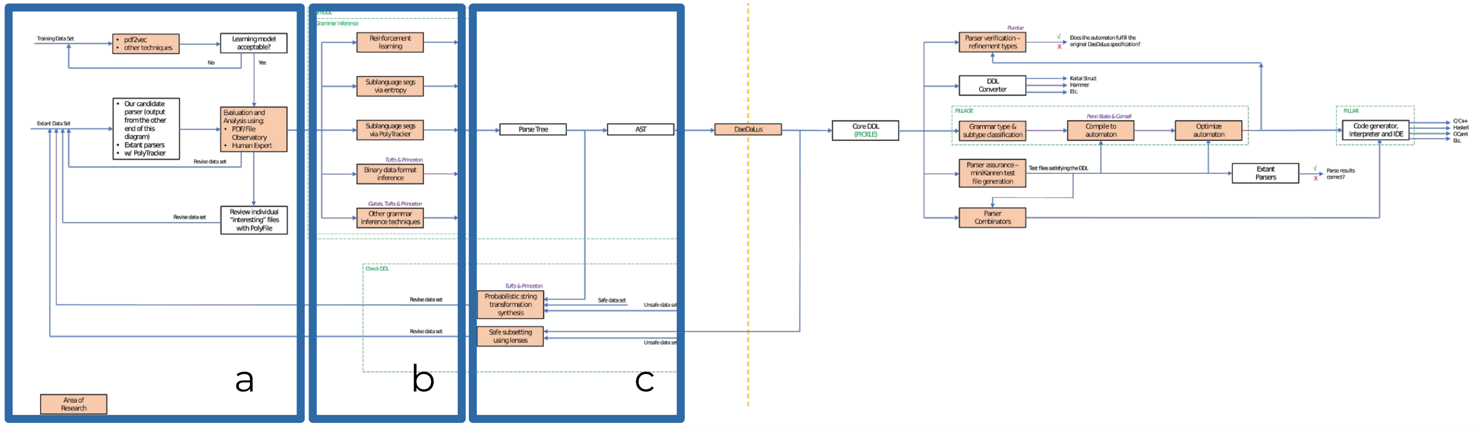
- Can affect data:
  - At rest (format ambiguity)
  - In use (parser vulnerabilities)
  - In transit (exfiltration)



ICARUS Project

- Pipeline and tools focused on developing methodologies for:
  a. Discovering and describing de facto data formats;
  b. Identifying grammars within a de facto format; and
  c. Translating from a de facto format to a safe subset.

Galois SafeDocs Structural Overview

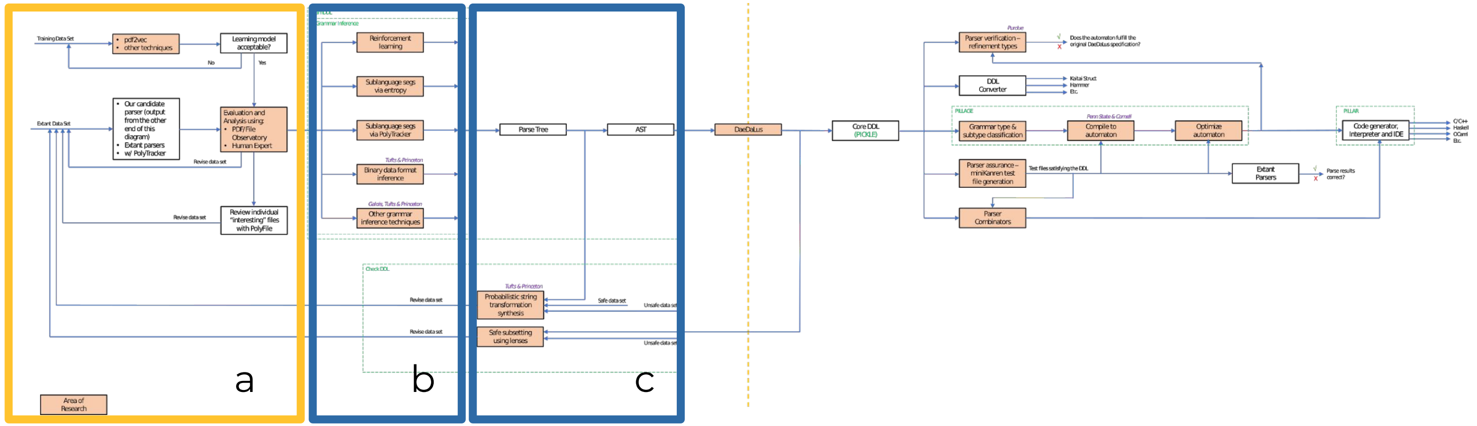


Training Data Set
pdf2vec
other techniques
Learning model acceptable?
No
Yes
Extant Data Set
Our candidate parser (output from the other end of this diagram)
Extant parsers
w/ PolyTracker
Evaluation and Analysis using:
PDF/File Observatory
Human Expert
Revise data set
Review individual "interesting" files with PolyFile
Revise data set
Area of Research
a
Grammar Inference
Reinforcement learning
Sublanguage segs via entropy
Sublanguage segs via PolyTracker
Tufts & Princeton
Binary data format inference
Galois, Tufts & Princeton
Other grammar inference techniques
Check DDL
Revise data set
Revise data set
b
Parse Tree
AST
Tufts & Princeton
Probabilistic string transformation synthesis
Safe data set
Unsafe data set
Safe subsetting using lenses
Unsafe data set
c
DaeDaLus
Core DDL
(PICKLE)
Purdue
Parser verification – refinement types
Does the automaton fulfill the original DaeDaLus specification?
DDL Converter
Kaitai Struct
Hammer
Etc.
PILLAGE
Grammar type & subtype classification
Penn State & Cornell
Compile to automaton
Optimize automaton
Parser assurance – miniKanren test file generation
Test files satisfying the DDL
Extant Parsers
Parse results correct?
Parser Combinators
PILLAR
Code generator, Interpreter and IDE
C/C++
Haskell
OCaml
Etc.

ICARUS Project

- Pipeline and tools focused on developing methodologies for:
  a. Discovering and describing de facto data formats;
  b. Identifying grammars within a de facto format; and
  c. Translating from a de facto format to a safe subset.

Galois SafeDocs Structural Overview



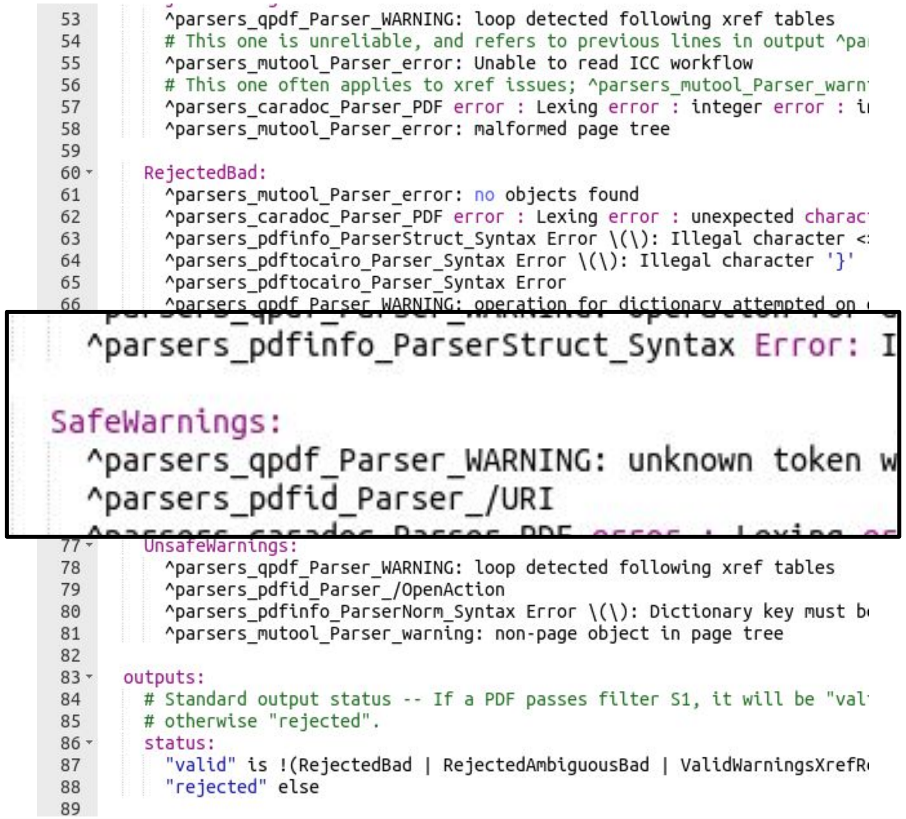
Training Data Set
pdf2vec
other techniques
Learning model acceptable?
No
Yes
Extant Data Set
Our candidate parser (output from the other end of this diagram)
Extant parsers
w/ PolyTracker
Evaluation and Analysis using:
PDF/File Observatory
Human Expert
Revise data set
Review individual "interesting" files with PolyFile
Revise data set
Area of Research
a
mDDL
Grammar Inference
Reinforcement learning
Sublanguage segs via entropy
Sublanguage segs via PolyTracker
Tufts & Princeton
Binary data format inference
Galois, Tufts & Princeton
Other grammar inference techniques
Check DDL
Revise data set
Revise data set
b
Parse Tree
AST
Tufts & Princeton
Probabilistic string transformation synthesis
Safe data set
Unsafe data set
Safe subsetting using lenses
Unsafe data set
c
DaeDaLus
Core DDL (PICKLE)
Purdue
Parser verification – refinement types
√
X
Does the automaton fulfill the original DaeDaLus specification?
DDL Converter
Kaitai Struct
Hammer
Etc.
PILLAGE
Grammar type & subtype classification
Penn State & Cornell
Compile to automaton
Optimize automaton
Parser assurance – miniKanren test file generation
Test files satisfying the DDL
Extant Parsers
√
X
Parse results correct?
Parser Combinators
PILLAR
Code generator, Interpreter and IDE
C/C++
Haskell
OCaml
Etc.

File Observatory

- Need to understand which collected files actually belong to de facto, vs malformed.
- Different parsers accept different files.
- Can learn a lot from stdout and stderr

Image from ImpulseCreative.com

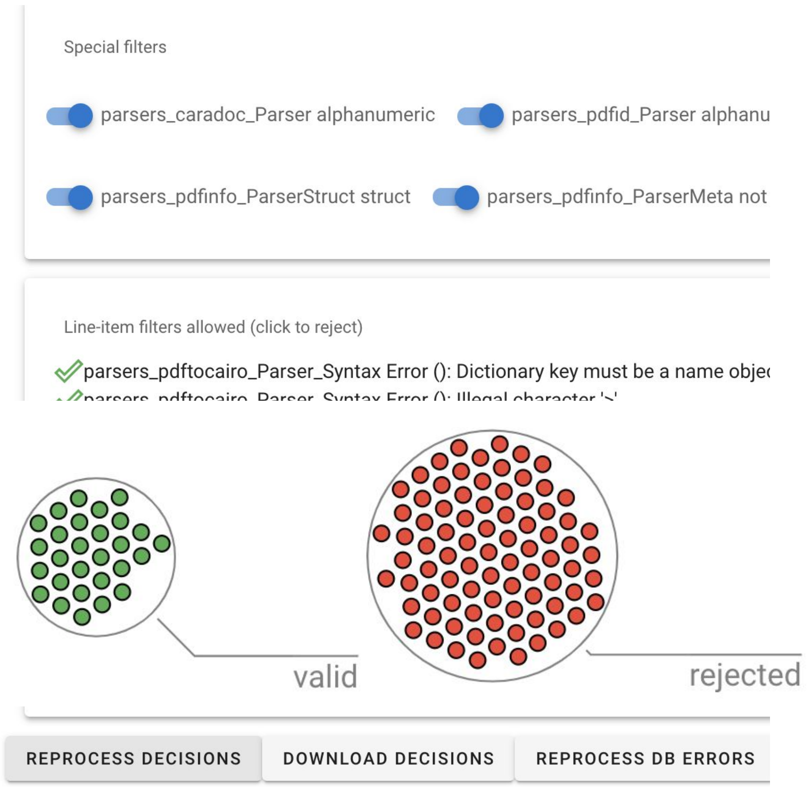
File Observatory



```
    ^parsers_qpdf_Parser_WARNING: loop detected following xref tables
    # This one is unreliable, and refers to previous lines in output ^pa
    ^parsers_mutool_Parser_error: Unable to read ICC workflow
    # This one often applies to xref issues; ^parsers_mutool_Parser_warn
    ^parsers_caradoc_Parser_PDF error : Lexing error : integer error : i
    ^parsers_mutool_Parser_error: malformed page tree

  RejectedBad:
    ^parsers_mutool_Parser_error: no objects found
    ^parsers_caradoc_Parser_PDF error : Lexing error : unexpected charac
    ^parsers_pdfinfo_ParserStruct_Syntax Error \(\): Illegal character <
    ^parsers_pdftocairo_Parser_Syntax Error \(\): Illegal character '}'
    ^parsers_pdftocairo_Parser_Syntax Error
    ^parsers_qpdf_Parser_WARNING: operation for dictionary attempted on
```

```
    ^parsers_pdfinfo_ParserStruct_Syntax Error: I

  SafeWarnings:
    ^parsers_qpdf_Parser_WARNING: unknown token w
    ^parsers_pdfid_Parser_/URI
```

```
  UnsafeWarnings:
    ^parsers_qpdf_Parser_WARNING: loop detected following xref tables
    ^parsers_pdfid_Parser_/OpenAction
    ^parsers_pdfinfo_ParserNorm_Syntax Error \(\): Dictionary key must b
    ^parsers_mutool_Parser_warning: non-page object in page tree

outputs:
  # Standard output status -- If a PDF passes filter S1, it will be "val
  # otherwise "rejected".
  status:
    "valid" is !(RejectedBad | RejectedAmbiguousBad | ValidWarningsXrefR
    "rejected" else
```



Special filters

parsers_caradoc_Parser alphanumeric
parsers_pdfid_Parser alphanu
parsers_pdfinfo_ParserStruct struct
parsers_pdfinfo_ParserMeta not

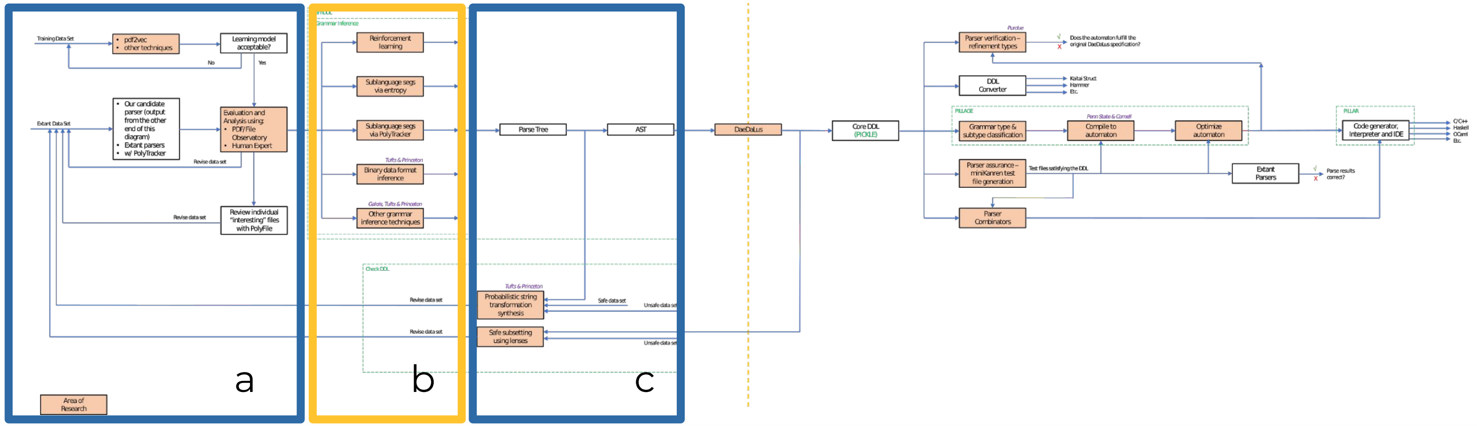
Line-item filters allowed (click to reject)

✓ parsers_pdftocairo_Parser_Syntax Error (): Dictionary key must be a name obje
✓ parsers_pdftocairo_Parser_Syntax Error (): Illegal character '>'

ICARUS Project

- Pipeline and tools focused on developing methodologies for:
  a. Discovering and describing de facto data formats;
  b. Identifying grammars within a de facto format; and
  c. Translating from a de facto format to a safe subset.

Galois SafeDocs Structural Overview



Training Data Set
pdf2vec
other techniques
Learning model acceptable?
No
Yes
Extant Data Set
Our candidate parser (output from the other end of this diagram)
Extant parsers
w/ PolyTracker
Evaluation and Analysis using:
PDF/File Observatory
Human Expert
Revise data set
Review individual "interesting" files with PolyFile
Revise data set
Area of Research
a
mDDL
Grammar Inference
Reinforcement learning
Sublanguage segs via entropy
Sublanguage segs via PolyTracker
Tufts & Princeton
Binary data format inference
Galois, Tufts & Princeton
Other grammar inference techniques
Check DDL
Revise data set
Revise data set
b
Parse Tree
AST
Tufts & Princeton
Probabilistic string transformation synthesis
Safe data set
Unsafe data set
Safe subsetting using lenses
Unsafe data set
c
DaeDaLus
Core DDL (PICKLE)
Purdue
Parser verification – refinement types
√
X
Does the automaton fulfill the original DaeDaLus specification?
DDL Converter
Kaitai Struct
Hammer
Etc.
PILLAGE
Penn State & Cornell
Grammar type & subtype classification
Compile to automaton
Optimize automaton
PILLAR
Code generator, Interpreter and IDE
C/C++
Haskell
OCaml
Etc.
Parser assurance – miniKanren test file generation
Test files satisfying the DDL
Extant Parsers
√
X
Parse results correct?
Parser Combinators

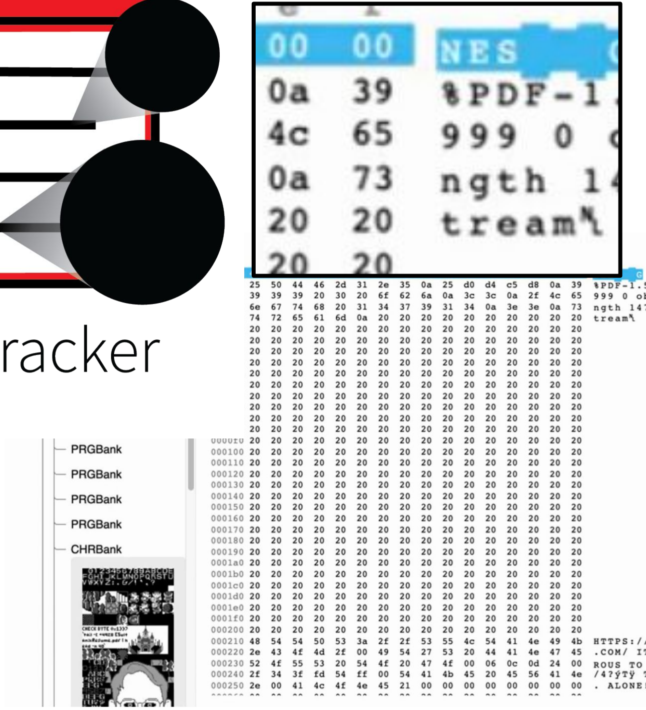
Identifying Sublanguages

- Large search space, unclear goals.
- Leverage known design principles.
- Three approaches surveyed:
  - **Taint Tracking**
  - **Entropy-based Methods**
  - **Reinforcement Learning**

Taint Tracking

- Instrument existing parsers to link program logic and file bytes.
- Taint forest used to track complex interdependencies (non-CFG).
- Improved by adding ground truth (PolyFile, PolyMerge).
- Differential analysis.
- Understanding existing vs building unified, de facto parser.

PolyTracker


```
00  00  NES
0a  39  %PDF-1
4c  65  999 0
0a  73  ngth 1
20  20  tream
20  20
```

```
       25 50 44 46 2d 31 2e 35 0a 25 d0 d4 c5 d8 0a 39  %PDF-1.
       39 39 39 20 30 20 6f 62 6a 0a 3c 3c 0a 2f 4c 65  999 0 o
       6e 67 74 68 20 31 34 37 39 31 34 0a 3e 3e 0a 73  ngth 14
       74 72 65 61 6d 0a 20 20 20 20 20 20 20 20 20 20  tream
       20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
       20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
       20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
       20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
       20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
       20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
       20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
       20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
       20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
       20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
0000f0 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
000100 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
000110 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
000120 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
000130 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
000140 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
000150 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
000160 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
000170 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
000180 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
000190 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
0001a0 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
0001b0 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
0001c0 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
0001d0 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
0001e0 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
0001f0 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
000200 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
000210 48 54 54 50 53 3a 2f 2f 53 55 4c 54 41 4e 49 4b  HTTPS:/
000220 2e 43 4f 4d 2f 00 49 54 27 53 20 44 41 4e 47 45  .COM/ I
000230 52 4f 55 53 20 54 4f 20 47 4f 00 06 0c 0d 24 00  ROUS TO
000240 2f 34 3f fd 54 ff 00 54 41 4b 45 20 45 56 41 4e  /4?ýTÿ
000250 2e 00 41 4c 4f 4e 45 21 00 00 00 00 00 00 00 00  . ALONE
```

Entropy-Based Methods

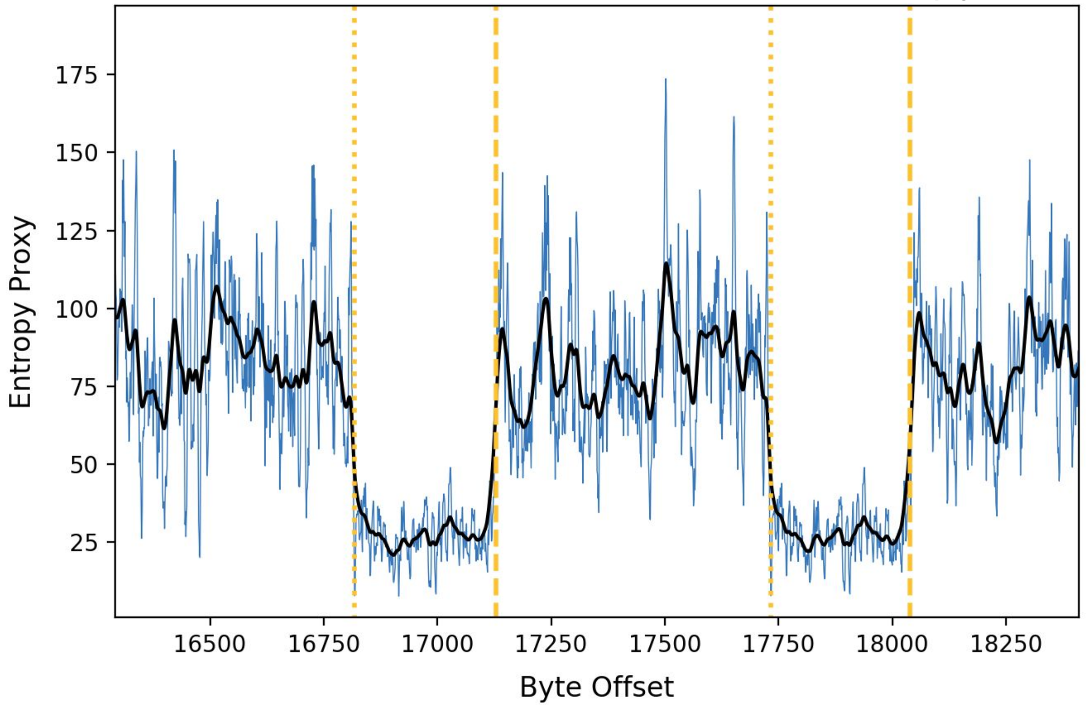
- Parserless
- Data fingerprints
- Example result: PDF streams

Byte values

$$\frac{\sum_{i=0}^{n-2} |S_i - S_{i+1}|}{n-1}$$

Window size $\longrightarrow n-1$

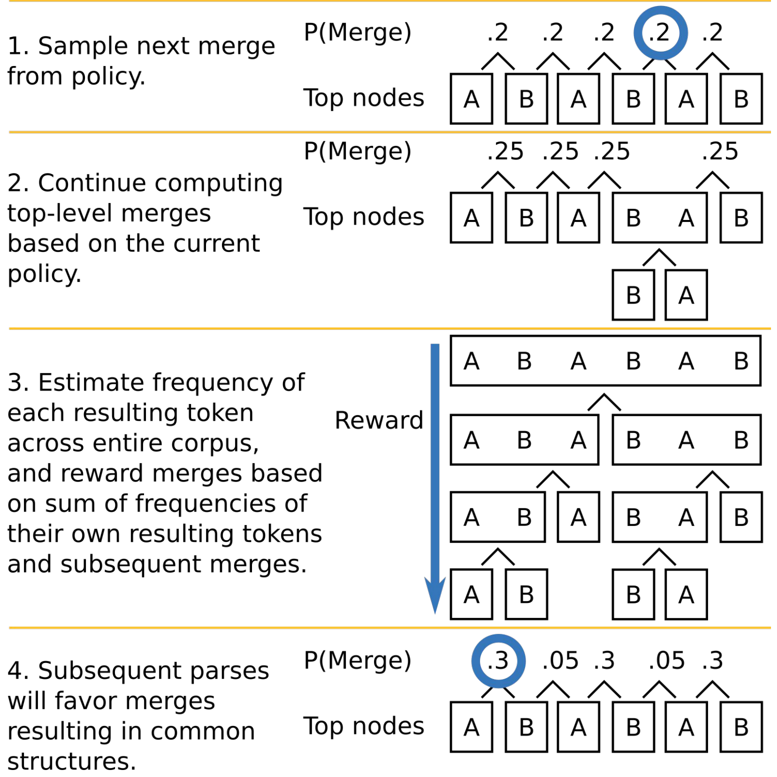
Position in File vs Measured Entropy



Grammar Inference via Reinforcement Learning

- State-of-the-art grammar inference is focused on NLP, not data (outside of LearnPADS).
- Our research has focused on bottom-up parsing algorithm, using RL+statistics to derive parsers.
- RL provides flexibility for grammars outside of CFGs.

Learning to parse (AB)* via RL, example string ABABAB



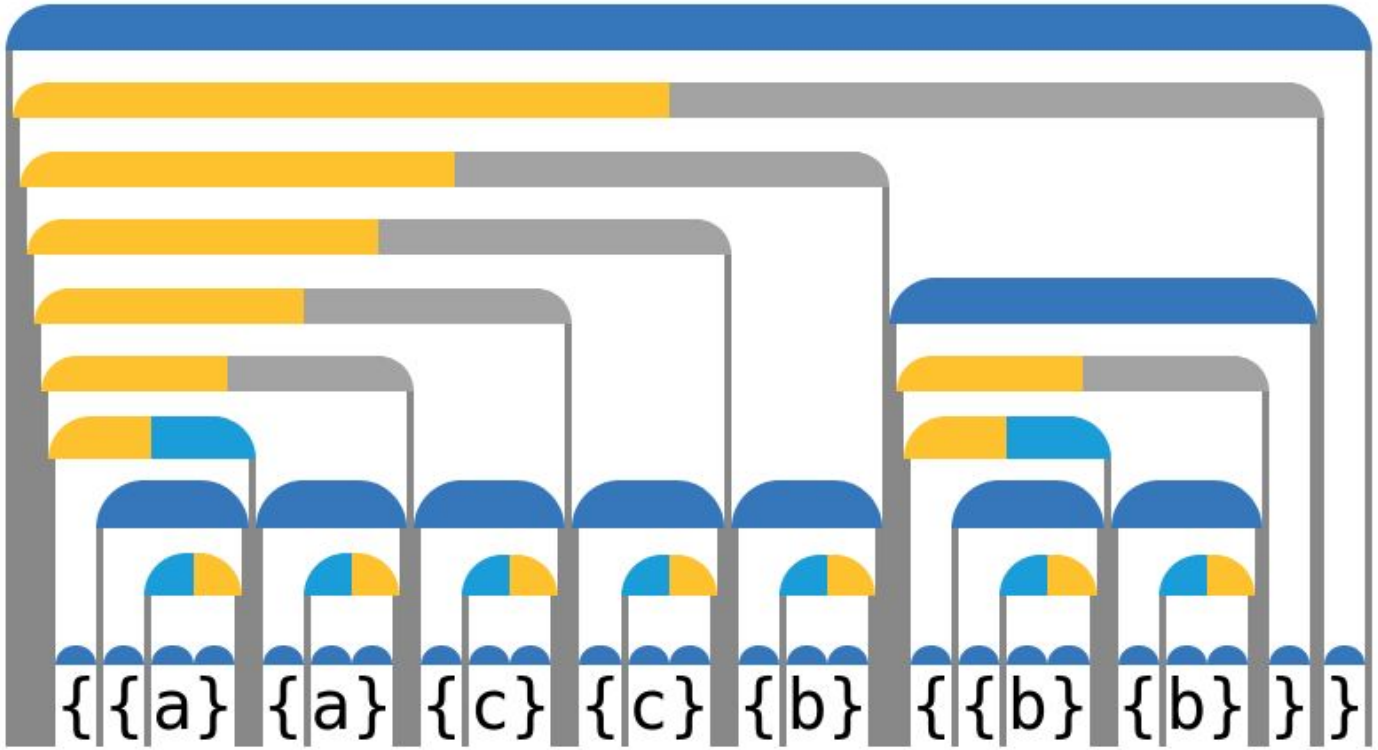
1. Sample next merge from policy.

2. Continue computing top-level merges based on the current policy.

3. Estimate frequency of each resulting token across entire corpus, and reward merges based on sum of frequencies of their own resulting tokens and subsequent merges.

4. Subsequent parses will favor merges resulting in common structures.

Grammar Inference via Reinforcement Learning

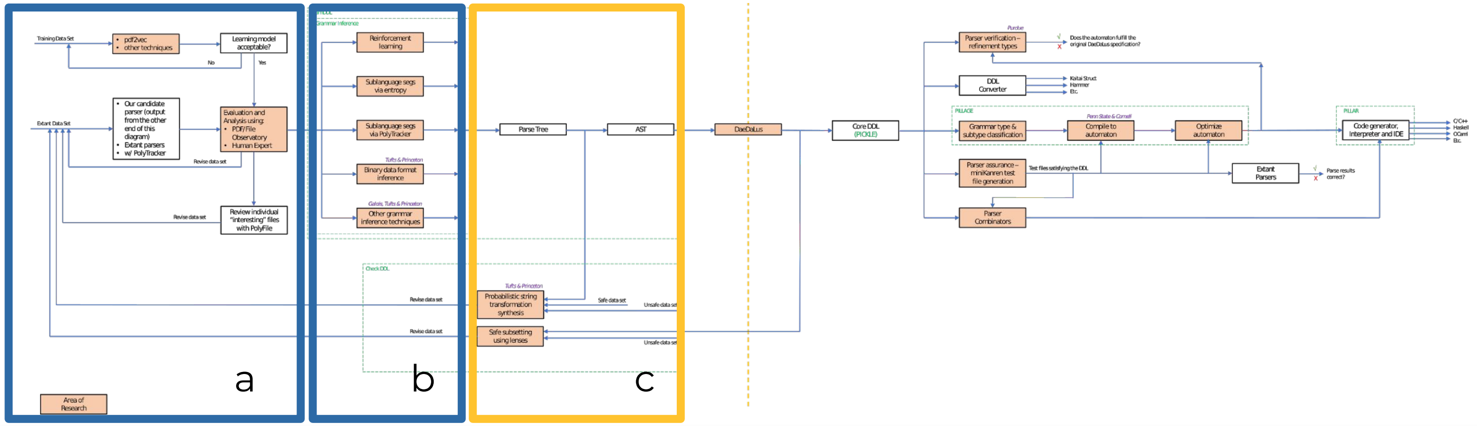


How might we handle e.g. streams?

ICARUS Project

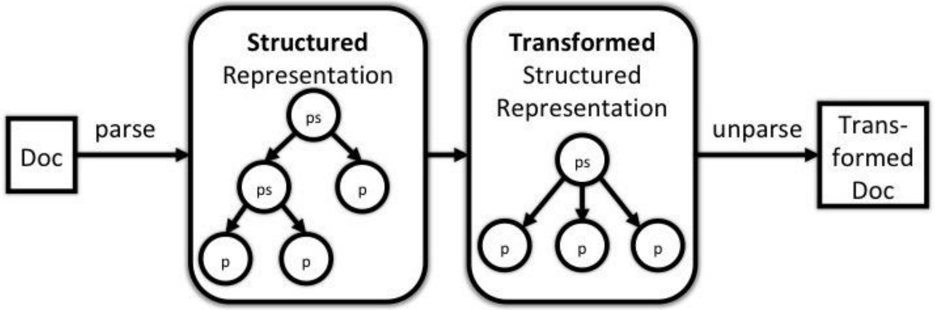
- Pipeline and tools focused on developing methodologies for:
  a. Discovering and describing de facto data formats;
  b. Identifying grammars within a de facto format; and
  c. Translating from a de facto format to a safe subset.

Galois SafeDocs Structural Overview


Training Data Set
pdf2vec
other techniques
Learning model acceptable?
No
Yes
Extant Data Set
Our candidate parser (output from the other end of this diagram)
Extant parsers
w/ PolyTracker
Evaluation and Analysis using:
PDF/File Observatory
Human Expert
Revise data set
Review individual "interesting" files with PolyFile
Revise data set
Area of Research
a
Grammar Inference
Reinforcement learning
Sublanguage segs via entropy
Sublanguage segs via PolyTracker
Tufts & Princeton
Binary data format inference
Galois, Tufts & Princeton
Other grammar inference techniques
Check DDL
Revise data set
Revise data set
b
Parse Tree
AST
Tufts & Princeton
Probabilistic string transformation synthesis
Safe data set
Unsafe data set
Safe subsetting using lenses
Unsafe data set
c
DaeDaLus
Core DDL (PICKLE)
Purdue
Parser verification – refinement types
Does the automaton fulfill the original DaeDaLus specification?
DDL Converter
Kaitai Struct
Hammer
Etc.
PILLAGE
Grammar type & subtype classification
Penn State & Cornell
Compile to automaton
Optimize automaton
Parser assurance – miniKanren test file generation
Test files satisfying the DDL
Extant Parsers
Parse results correct?
Parser Combinators
PILLAR
Code generator, Interpreter and IDE
C/C++
Haskell
OCaml
Etc.

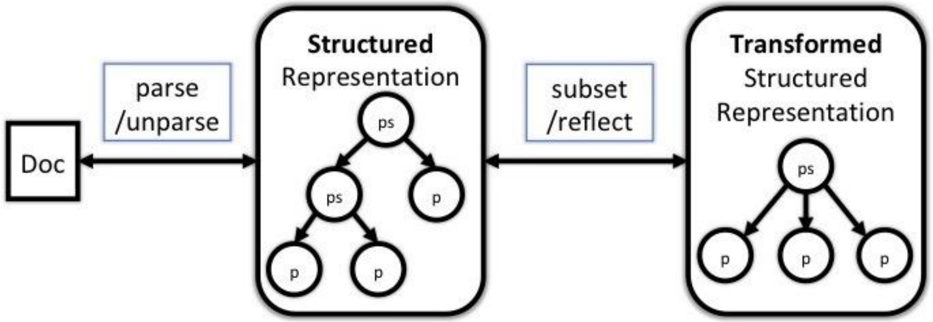
Safe Subsetting

- Desire to accept as much of de facto population as possible, while providing safety.


Doc
parse
Structured Representation
ps
ps
p
p
p
Transformed Structured Representation
ps
p
p
p
unparse
Trans-formed Doc

- Traditional subsetting.

- Bidirectional programming approach.


Doc
parse /unparse
Structured Representation
ps
ps
p
p
p
subset /reflect
Transformed Structured Representation
ps
p
p
p

Safe Subsetting

1. << /Size 14 /Version 5 >>
2. << /Size 14 /Version 5 /Version 6 >>
3. << /Size null /Version 5 >> == << /Version 5 >>
4. << /Size 14 /Version 5 /Version null >> == **??**
5. Lens ordering:
   - removeNullEntries . rejectDictionaryDups
   - rejectDictionaryDups . removeNullEntries

Conclusions and Future Work

- ICARUS Toolchain being assembled to understand and secure de facto formats through:
  - Leveraging existing parsers,
  - Inferring de facto grammar,
  - Safe subsetting.