Anomaly Detection with Neural Parsers That Never Reject

Alexander Grushin and Walt Woods
Galois, Inc.

Motivation

- **Formally defining the language for inputs that an application accepts (LangSec)**
  - E.g., in an enterprise system, do applications have consistent behavior for a given input?
  - For most applications, no such language is defined, but could the rules of the language be inferred from example inputs?
- **Understanding how an existing format is *really* used (DARPA SafeDocs)**
  - E.g., There are many parsers for the Portable Document Format (PDF) format, which interpret the format in different ways
    - Features are added
    - Bugs may exist
  - Again, can we infer the grammar from examples of PDF documents?

Related Work – Recurrent Neural Networks

- **Trained to estimate the probability of a *token* (~character) in a *sentence* (~string), given other tokens**
  - E.g., how likely are we to observe a '`}`' at the end of `{x}`?
- **Can capture *data type recurrency* in non-regular languages**
- **If the probabilities are sufficiently high, the sentence is accepted**
- **If a sentence is in the correct format, but has a *high-entropy region*, then it may be rejected**

JSON Object

High-Entropy Region

```
{ "a": {"b": { "c": "d" } } }   { "y": "PxdGmfn3ea" }
```

JSON Object

Related Work – Autoencoders or Transformers

- **Used to perform *constituency parsing*, i.e., to merge pairs of related, adjacent tokens in a sentence into *atoms*, which are recursively merged into higher-level atoms**
  - E.g., in the sentence `I am`:
    - `'am' -> 'a' + 'm'`
    - `' am' -> ' ' + 'am'`

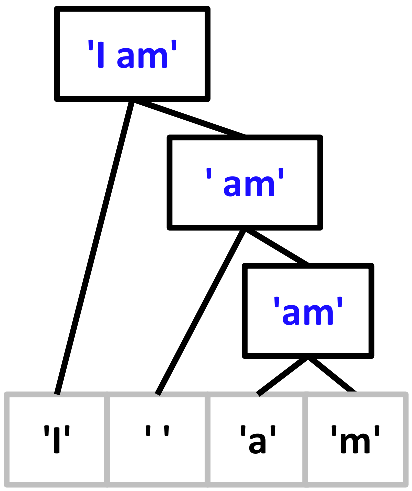
Merges can be expressed as production rules

- **A binary parse tree can be produced for any sentence, even if it contains high-entropy regions**
- **Data type recurrency is not captured**

Atom 2

```
{ "a": {"b": { "c": "d" } } }
```

Atom 1



Related Work – Reinforcement Learning (RL-GRIT)

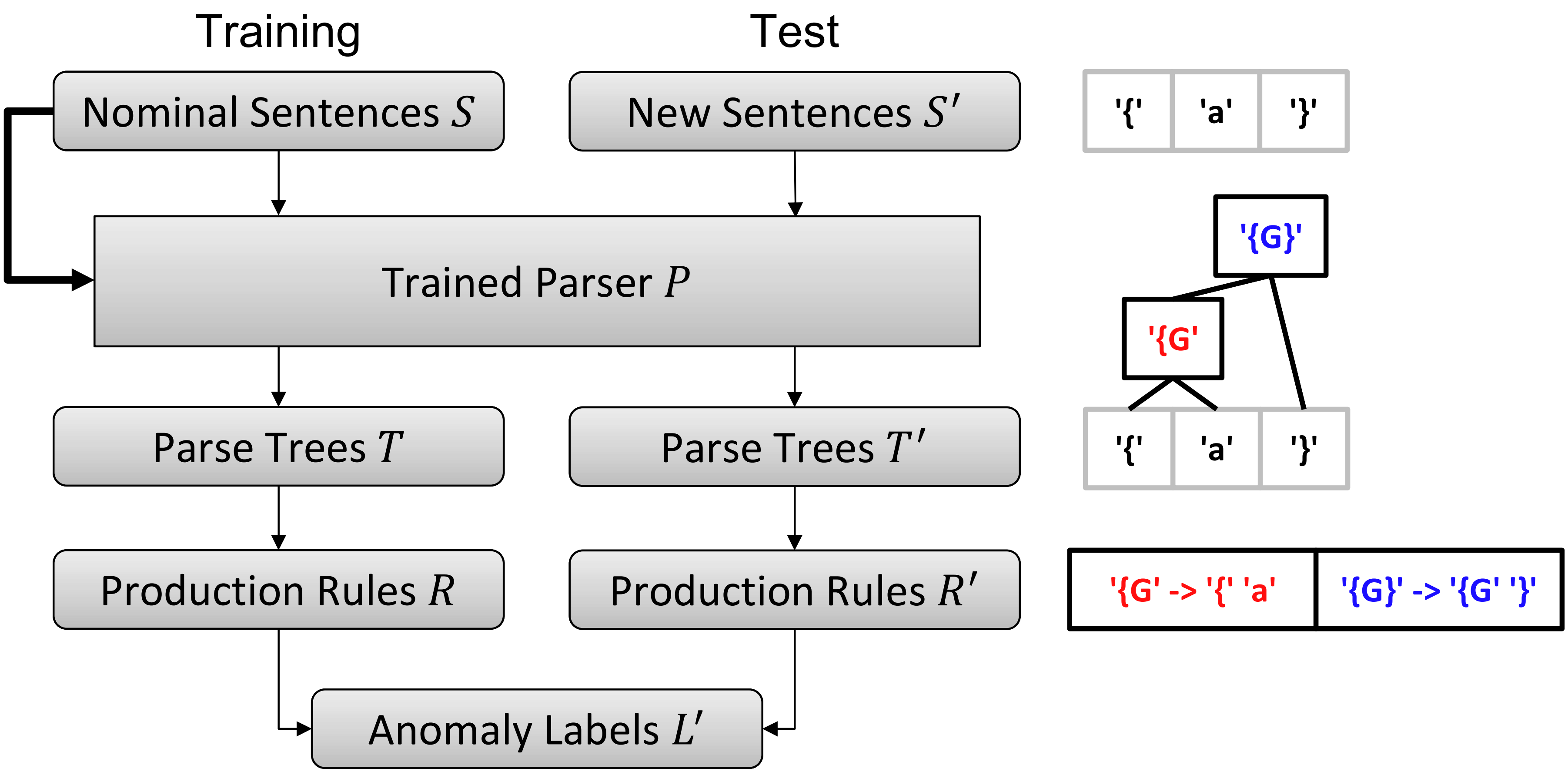
- **Developed at Galois; presented at LangSec 2020 and 2021**
- **Makes use of reinforcement learning**
  - Each merge is treated as an *action*
  - The set of current atoms is the *observation*
  - A parser maps observations to actions to maximize *reward*
- **This formulation allows for a broader set of actions, which allow for data type recurrency:**
  - Regular merge: `'am' -> 'a' + 'm'`
  - Anchored merge: `'a' -> 'a' + 'm'` (~ the `*` operator)
  - Subgrammar merge: `'aG' -> 'a' + 'm'` (~ the `|` operator)
    `'aG' -> 'a' + 'n'`
- **Limitation: because any sentence can be parsed, how can we determine if a sentence is *anomalous* for some format?**

Contributions

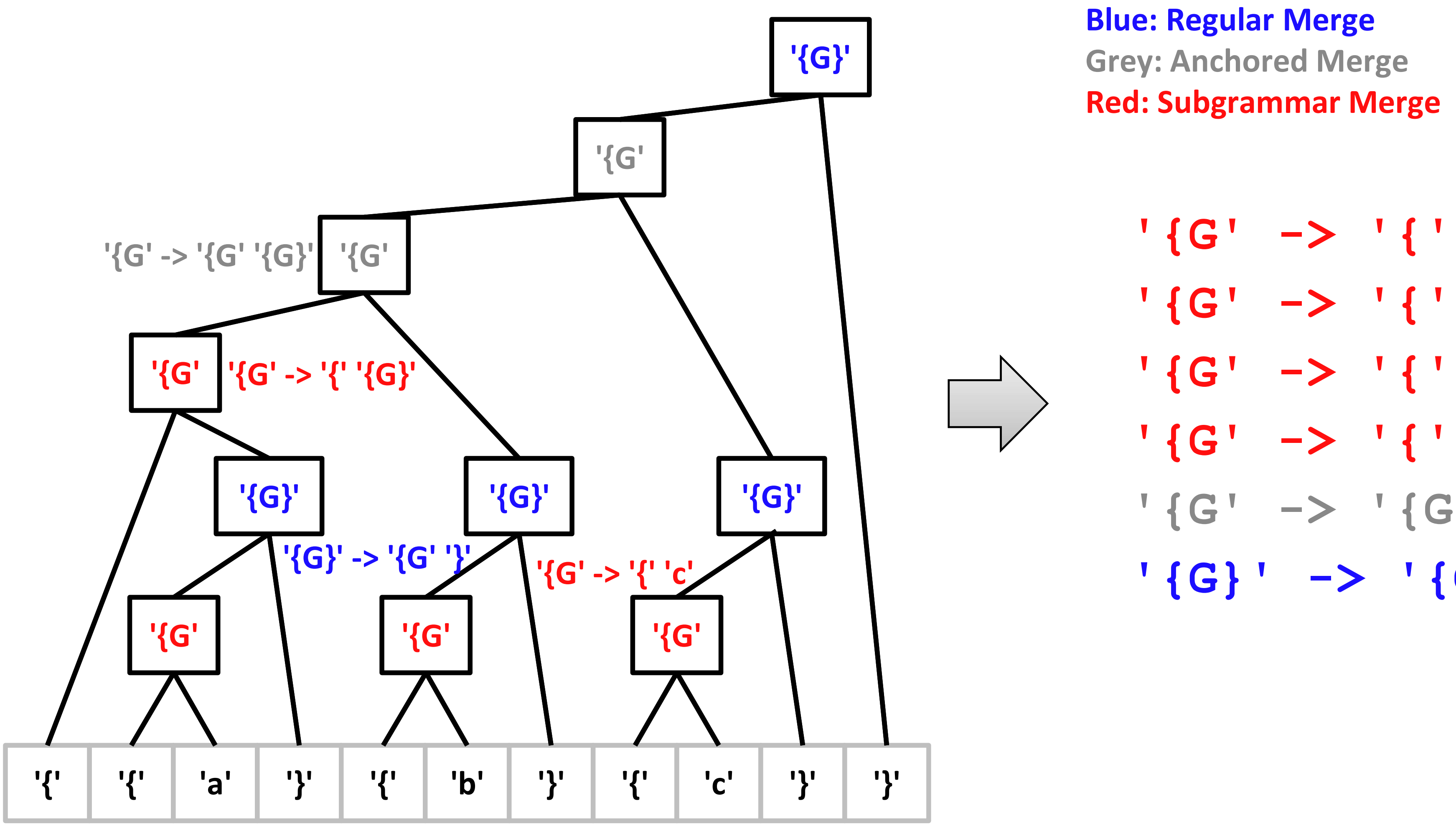
- **We extend RL-GRIT approach by:**
  - Extracting a grammar from the parser
  - Using the grammar to identify anomalies in sentences
  - Distinguishing between anomalies and high-entropy regions
- **To our knowledge our approach is the first to demonstrate anomaly detection in unknown formats with data type recurrency or high-entropy regions**

Approach	Data Type Recurrency	High-Entropy Regions	Anomaly Detection
Recurrent Neural Networks	√	?	√
Autoencoders or Transformers	X	√	X
Reinforcement Learning (RL-GRIT)	√	√	X
This Study	√	√	√

Approach Overview



Production Rule Extraction (Training)

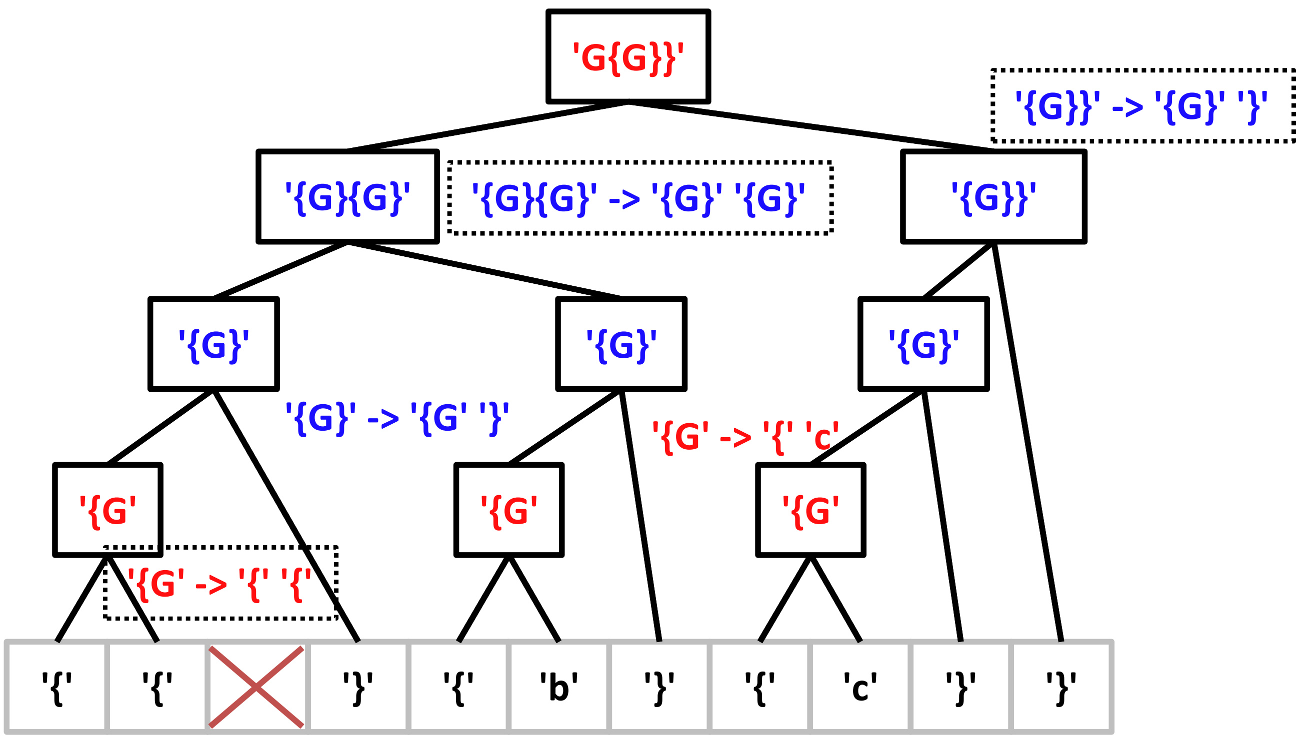



```
'{G' -> '{' 'a'
'{G' -> '{' 'b'
'{G' -> '{' 'c'
'{G' -> '{' '{G}'
'{G' -> '{G' '{G}'
'{G}' -> '{G' '}'
```

`S -> '{' ('a' | 'b' | 'c' | S+) '}'` (*Simple-JSON* grammar)

Anomaly Detection and Localization (Test)

- A rule *covers* a token if the token's node is a child of the rule's node
- Tokens that are covered by unexpected rules are labeled as potentially anomalous

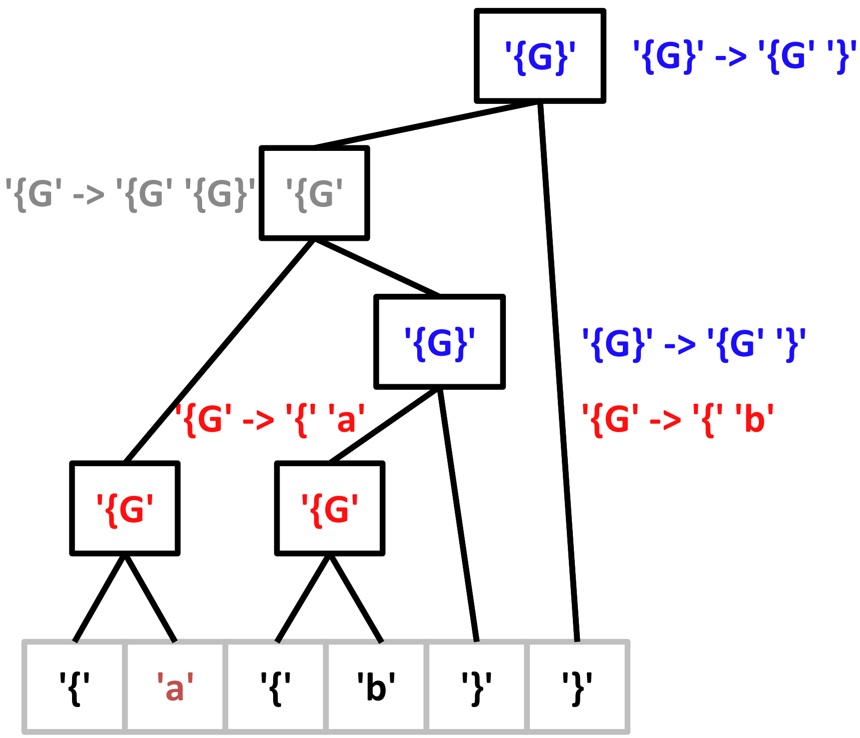



```
'{G' -> '{' 'a'
'{G' -> '{' 'b'
'{G' -> '{' 'c'
'{G' -> '{' '{G}'
'{G' -> '{G' '{G}'
'{G}' -> '{G' '}'
```

Unexpected rules exist; the sentence is labeled as anomalous

Localization Rate = $^1/_1$ Localization Ratio = $^3/_{10}$

Precedence Constraints

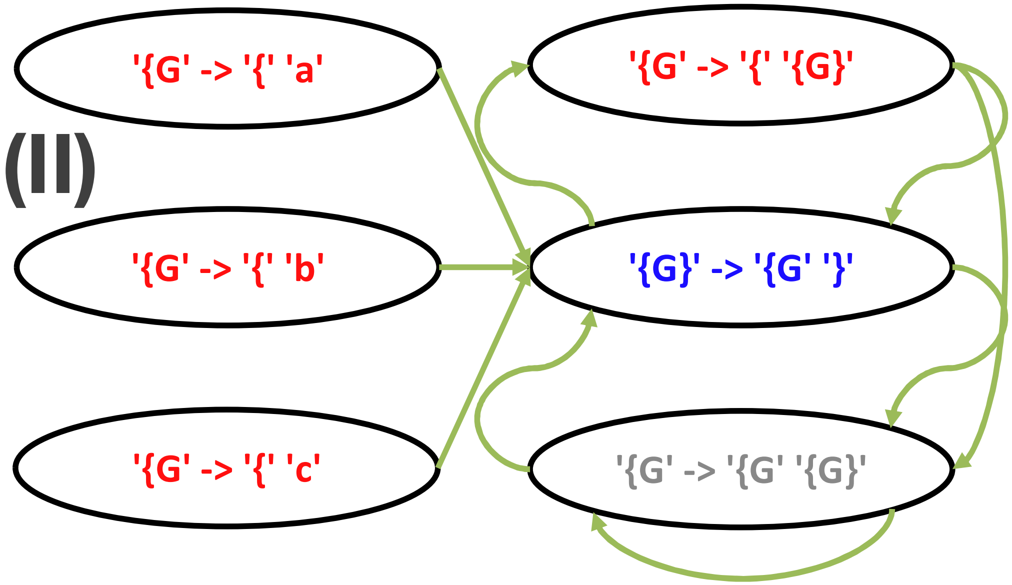


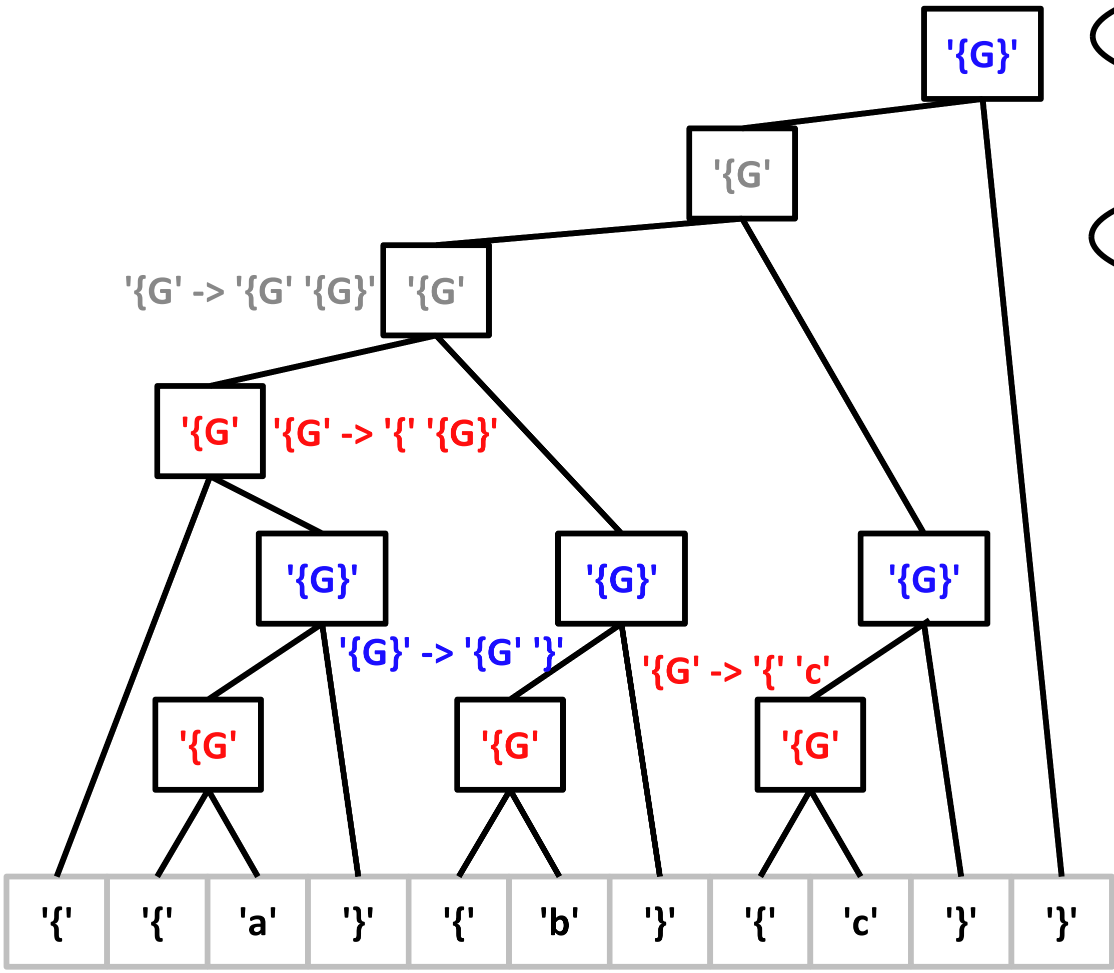

```
'{G' -> '{' 'a'
'{G' -> '{' 'b'
'{G' -> '{' 'c'
'{G' -> '{' '{G}'
'{G' -> '{G' '{G}'
'{G}' -> '{G' '}'
```

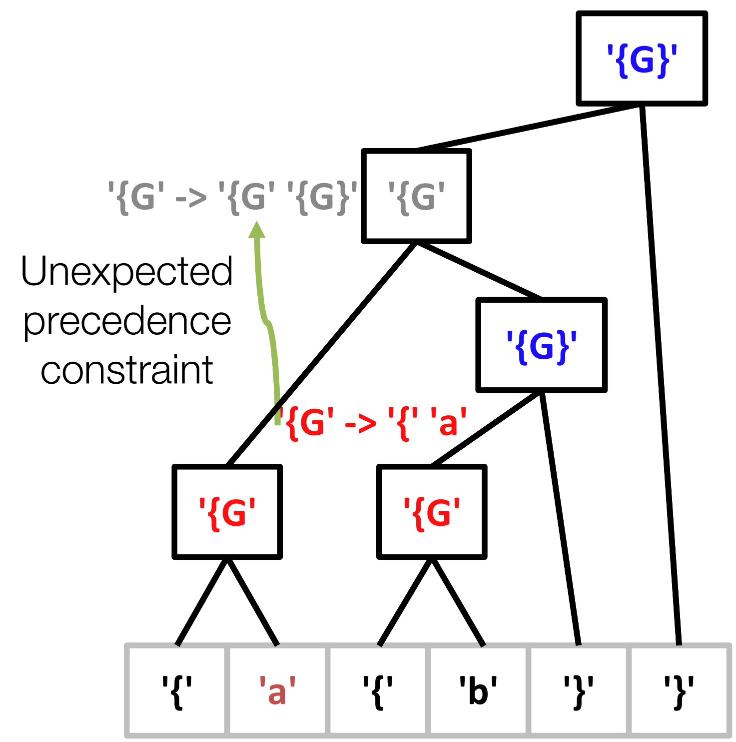
The sentence is anomalous, but there are no unexpected rules!

Issue: many identical atoms due to anchor and subgrammar merges

Precedence Constraints (II)







Sentence Simplification

- **Consider the *Key-List* grammar, containing sentences such as `/cjc /i /sp`**
  - Because each key contains a high-entropy region, the language cannot be described by a parsimonious set of rules
- **We can *simplify* Key-List sentences as follows:**
  - Train a parser, generate parse trees, and extract rules
  - Keep only the rules that occur frequently; e.g.:
    - `'/' -> ' ' '/'`
    - `'/' -> '/' '/'`
  - Find high-entropy regions, using the frequent rules
    - Topological Approach: apply anomaly localization, to see which tokens are covered by rules other than the above
    - Symbolic Approach: if a token does not exist on the right side of the above rules, label it as high-entropy

Sentence Simplification (II)

- **Continuing from above:**
  - Replace each high-entropy region with a *high-entropy token* `'&'`, e.g.:
    - `/cjc /i /sp` → `/& /& /&`
    - `/cjc i /sp` → `/& & /&`
- **Now, apply the pipeline a second time, to the simplified sentences**
  - Train a parser, generate parse trees, and extract rules
  - Apply anomaly detection and localization

Evaluation: Simple-JSON

Data Subset	Nominal Sentences	Anomalous Sentences
Training	120	0
Rule Extraction	100	0
Validation	100	0
Evaluation	100	100 × 3 anomaly types

Ran 30 trials, with a different parser trained each time (training is stochastic)

- Considered the 18 trials where the validation false positive was 0%
- For the evaluation set, the false positive rate was also 0% across the 18 trials, and 6.9% across all 30 trials

Anomaly	True Positive Rate	Localization Rate	Localization Ratio
Deleted Bracket	100.0%	18.2%	9.2%
Deleted Letter	100.0%	100.0%	12.3%
Inserted Letter	100.0%	62.1%	7.5%

Localization Rate: high is good

Localization Ratio: low is good

Evaluation: Key-List and Simple-JSON-Stream

- **Applied the symbolic simplification approach to Key-List sentences, with a space or a forward slash deleted in some sentences; validation was not done**
  - False positive rate: 0%
  - True positive rate: 87%

- **We also applied topological simplification to *Simple-JSON-Stream* sentences:**
  - `{hfsawpl{{a}}ygictfxk` → `&{{a}}&` (correct)
  - `v{uptffaxlnnjh{{b}{b}{a}}plvalinjhxrmcjb` → `&{{&{b}{a}}&` (incorrect)
    - RL was somewhat sensitive to high-entropy regions, resulting in suboptimal rule sets, and simplification errors

Conclusions and Future Work

- **We extended RL-GRIT, an RL-based parser learning approach to extract grammars from the parser, and to use these grammars for detecting and localizing anomalies**
- **We successfully applied the approach to a format with data type recurrency, and a format with high-entropy regions**
- **The RL algorithm is being further improved, to extend the approach to more complex:**
  - Formats (e.g., Simple-JSON-Stream, PDF dictionaries)
  - Anomalies (e.g., those involving changes in multiple tokens)
- **It is also of interest to characterize the expressive power of the rule / precedence constraint representation**
  - Open question: can the resulting representation describe any context-free language?

Acknowledgments

- DARPA
- The SafeDocs team
- Richard Jones
- Julien Vanegue
- Anonymous reviewers

Bibliography

- **Recurrent Neural Networks**
  - D. M. Yellin and G. Weiss, “Synthesizing Context-free Grammars from Recurrent Neural Networks (Extended Version),” arXiv:2101.08200, 2021
  - B. Barbot, B. Bollig, A. Finkel, S. Haddad, I. Khmelnitsky, M. Leucker, D. Neider, R. Roy, and L. Ye, “Extracting context-free grammars from recurrent neural networks using tree-automata learning and a* search,” in Proceedings of the Fifteenth International Conference on Grammatical Inference, 2021

- **Autoencoders or Transformers**
  - A. Drozdov, P. Verga, Y.-P. Chen, M. Iyyer, and A. McCallum, “Unsupervised labeled parsing with deep inside-outside recursive autoencoders,” in Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), 2019
  - A. Drozdov, S. Rongali, Y.-P. Chen, T. O’Gorman, M. Iyyer, and A. McCallum, “Unsupervised parsing with s-diora: Single tree encoding for deep inside-outside recursive autoencoders,” in Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), 2020
  - Y.-S. Wang, H.-Y. Lee, and Y.-N. Chen, “Tree transformer: Integrating tree structures into self-attention,” arXiv:1909.06639, 2019

- **Reinforcement Learning**
  - S. Cowger, Y. Lee, N. Schimanski, M. Tullsen, W. Woods, R. Jones, E. Davis, W. Harris, T. Brunson, C. Harmon et al., “Icarus: Understanding de facto formats by way of feathers and wax,” in 2020 IEEE Security and Privacy Workshops (SPW), 2020
  - W. Woods, “RL-GRIT: Reinforcement learning for grammar inference,” in 2021 IEEE Security and Privacy Workshops (SPW), 2021