

Paper Title: Grammar-Based Anomaly Detection of Microservice Systems Execution Traces
Paper Link: <https://doi.org/10.1145/3629527.3651844>

1 Summary

1.1 Motivation

Microservice architectures generate complex distributed traces difficult to analyze manually. Existing ML-based anomaly detection methods achieve high accuracy but require significant computational resources, training time, and operate as black boxes lacking interpretability. This paper aims to provide an efficient, interpretable alternative using grammar-based pattern extraction that reduces training time while maintaining competitive accuracy.

1.2 Contribution

The paper introduces a novel grammar-based approach combining SAX (Symbolic Aggregate approXimation) encoding with Sequitur grammar induction for anomaly detection. Key contributions: (1) converting numerical latency sequences into symbolic patterns, (2) extracting human-readable Context-Free Grammar rules characterizing normal behavior, (3) achieving ~ 15 seconds faster training than Logistic Regression with only $\sim 5\%$ effectiveness loss, enabling operators to understand detection decisions through parse trees.

1.3 Methodology

Three-stage pipeline: (1) SAX encoding discretizes latency values into symbolic alphabet (e.g., 'a' for low, 'c' for high latency) using 5 bins, (2) Sequitur algorithm induces Context-Free Grammar from anomalous traces, building production rules that capture recurring patterns, (3) membership testing classifies new traces—if SAX-encoded trace belongs to anomaly grammar language $L(G)$, it's anomalous; otherwise normal. Evaluated on E-Shopper and Train-Ticket benchmarks, compared against Logistic Regression baseline.

1.4 Conclusion

The grammar-based method achieves competitive effectiveness (accuracy, precision, recall) compared to Logistic Regression while requiring significantly less training time (13-14 seconds faster). The approach provides interpretable parse trees showing where anomalies occur. However, it relies solely on latency-derived syntactic patterns and uses a naive grammar induction algorithm (Sequitur), leaving room for optimization with more sophisticated algorithms like ARVADA.

2 Limitations

2.1 First Limitation

Single Feature Type: Uses only latency-based temporal features, ignoring service topology, call graphs, and structural dependencies. Microservice anomalies often manifest in architectural patterns (cascading failures, circular dependencies, bottleneck services), which grammar rules from timing sequences alone cannot capture. Missing spatial/structural context limits detection of topology-dependent anomalies.

2.2 Second Limitation

Limited Root Cause Analysis: While parse trees show that a pattern violated grammar rules, they don't identify which specific microservices caused the anomaly or why. The approach detects "something is wrong" but provides minimal diagnostic information for operators to pinpoint root causes. Sequitur's naive grammar induction also produces large, non-optimal grammars, reducing interpretability despite theoretical advantages.

3 Synthesis

The grammar approach provides a foundation for efficient, interpretable anomaly detection. My project **xHybrid** extends this by addressing both limitations: (1) combining grammar features with Graph Neural Networks (capturing service topology/dependencies) and LSTM (modeling temporal patterns), (2) implementing multi-level explainability—grammar rules show pattern violations, GNN attention weights highlight critical services, SHAP values quantify feature contributions. This hybrid architecture balances the accuracy-interpretability-efficiency trade-off, achieving better detection while maintaining computational efficiency and providing actionable insights for operators.

Tools: PyTorch, PyTorch Geometric, saxpy, sksequitur, SHAP, NetworkX.