SIGMA: Systematic Island Grammar forMation Approach Merging Grammars

Isaac Griffith and Rosetta Roberts

Empirical Software Engineering Laboratory
College of Science and Technology, Idaho State University

Idaho State University | Computer Science

TRIO RONALD E. MCNAIR POST-BACCALAUREATE ACHIEVEMENT PROGRAM

Introduction

Motivation—Many modern systems are written in multiple programming languages which are difficult to analyze. One approach to ease this process is to use multilingual island grammar parsers. In order to automate the process of creating multilingual island grammar parsers, one needs to automate merging grammars in a way that decreases their complexity and maintenance effort.
Research Goal—We used the Goal-Question-Metric (GQM) paradigm [1] to form the following research goal: Develop an automated approach for merging components of a grammar to facilitate the construction of both the island and water components in an island grammar in order to reduce both the overall initial effort required in creating such grammars and the effort required to maintain such grammars.
Research Question—To evaluate our automated approach, we formed two research questions.

- What is the effect that this process has on the effort of the merged grammars? - It is expected that this process decreases the maintenance effort.
- What is the effect that this process has on the complexity of the merged grammars? - It is expected that this process decreases the complexity.

Approach

Steps

1. Parse Grammars
2. Trivally Merge Grammars
3. Normalize Grammar
4. Measure Production Similarities
5. Merge Most Similar Productions
6. Repeat Steps 3–5 Until Max Similarity is Below a Threshold
7. Output Grammars

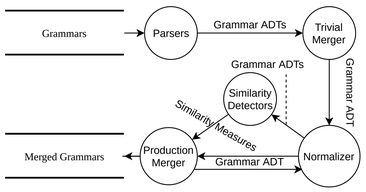

Fig. 1: Merge Process

Data Model

SIGMA parses input grammars into the object oriented data model shown in figure 2. The model is created by visiting each node of the grammar's abstract syntax tree. The grammar is recreated by visiting each object of the model.

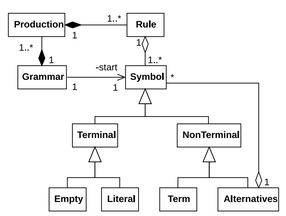

Fig. 2: Data Model

Measuring Production Similarity

The similarity of productions P_a, P_b that both expand to a sequence of symbols $p_1 p_2 \ldots$ is calculated by finding the Longest Common Subsequence (LCS) between the productions and then using the following formula.

$$\frac{2|LCS(P_a, P_b)|}{|P_a| + |P_b|}$$

The similarity of productions P_a, P_b that both expand to one of several symbols $p_1|p_2|\ldots$ is calculated by finding the number of shared symbols and then using the following formula.

$$\frac{2|P_a \cup P_b|}{|P_a| + |P_b|}$$

Normalization

SIGMA normalizes grammars so that all rules match one of two forms:

$$P_1 \rightarrow \texttt{A`a'B}$$
$$P_2 \rightarrow \texttt{A|`a'|B}$$

Experimental Design

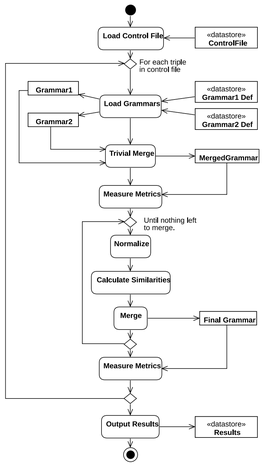

Fig. 3: Data Collection Process

- One Experiment for Each of ∆HAL and ∆MCC.
- 3*5 Factorial Design With 5 Repetitions
- Experimental Units — To select our experimental units, we split grammars from the ANTLR4[a] repository into 3 sizes, selected 12 grammars from each size category, and selected 50 unordered pairs of grammars from each size category.
- Threshold From Step 6 of Approach. We used 5 different levels of our threshold: .01, .25, .5, .75, 1.0. A threshold of 1.0 was our control.
- Experimental Measures. From [2].

PROD Number of Productions. Measure of Size of Grammars.
∆HAL Amount Halstead Effort Decreased. Measure of the Maintainability of Grammars.
∆MCC Amount Cylomatic Complexity Decreased. Measure of Grammar Complexity.

- Analysis
  - Permutation F-Test
  - Jonchheere-Terpstra Test
  - Steel Test

[a] https://github.com/antlr/grammars-v4

Results

∆HAL

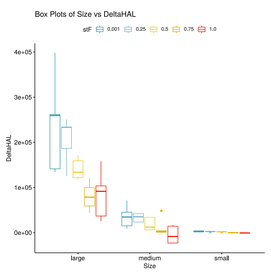

Fig. 4: ∆HAL Box Plot

∆MCC

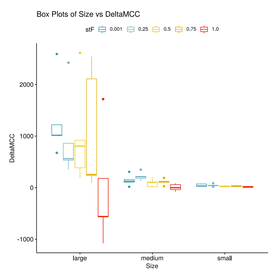

Fig. 6: ∆MCC Box Plot

Interaction

- F — 5.098
- p — 7.31e-05
- Influential primarily at control

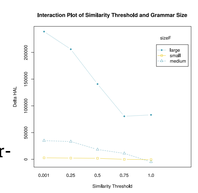

Fig. 5: ∆HAL Interaction Plot

1. Perm. F-Test — F: 9.569, p: 4.73e-06
2. JT Test — Statistic: 767, p: 6e-4
3. Steel Test — p:

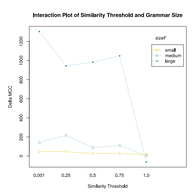

Fig. 7: ∆MCC Interaction Plot

- F —
- p —
- Influential primarily at control

1. Perm F-Test — F: , p:
2. JT Test — Statistic: , p:
3. Steel Test — p:

Discussion

- Our experiment found that SIGMA's merging process reduced the complexity and the maintainability effort of grammars regardless of their size and the threshold used. This is confirmed by the Steel Test.
- As the threshold decreases, complexity and maintainability effort decreased as confirmed by the JT Test.
- There were significant interactions between the threshold and size. However, these were only present at our control.

Threats to Validity

- Internal — We selected grammars from only the ANTLR4 grammar respository. Therefore our conclusions cannot necessarily be generalized past grammars in this repository.
- External — Our experimental units lacked grammars in formats commonly used in industry such as TXL, SDF, etc.
- Conclusion/Construct — None

Conclusions

We sucessfully created a tool that merges programming language grammars. This tool was demonstrated to decrease the maintainability effort and complexity of grammars. Because this merging process has the qualities that we desire, it can be used as part of a larger process for automatically creating island grammars for multilingual parsing.

Future Work

Possible directions for future work include the following. We intend to investigate the normalization process to ensure each step is necessary. We plan to evaluate this method on the larger collection of grammars GrammarZoo[3]. We also intend to integrate our approach with current approaches of creating tolerant grammars[4]. Finally we intend to augment our process with additional algorithms to allow it to automatically construct island grammars for multilingual parsing.

References

[1] V. R. B. G. Caldiera and H. D. Rombach, "The goal question metric approach," Encyclopedia of software engineering, pp. 528–532, 1994.

[2] J. F. Power and B. A. Malloy, "A metrics suite for grammar-based software," Journal of Software Maintenance and Evolution: Research and Practice, vol. 16, no. 6, pp. 405–426, Nov. 2004.

[3] V. Zaytsev, "Grammar Zoo: A corpus of experimental grammarware," Science of Computer Programming, vol. 98, pp. 28–51, Feb. 2015.

[4] S. Klusener and R. Lammel, "Deriving tolerant grammars from a base-line grammar," in International Conference on Software Maintenance, IEEE, 2003, pp. 179–188.

Acknowledgements

This research is supported by funding from the Ronald E. McNair Post Baccalaureate Achievement Program at Idaho State University, which is sponsored by the Department of Education (P217A170169).