SIGMA - Normalization

Isaac Griffith and Rosetta Roberts
Empirical Software Engineering Laboratory
Informatics and Computer Science
Idaho State University
Pocatello, Idaho, 83208
Email: {grifisaa@isu.edu, roberose@isu.edu}

***Abstract*—Introduction:**
Objective:
Methods:
Results: What are the main findings? Practical implications?
Limitations: What are the weaknesses of this research?
Conclusions: What is the conclusion?

***Index Terms*—Island Grammars, Automated Grammar Formation, Software Language Engineering**

I. Introduction

REWORD

Modern software development practice has led to the creation of software systems using multiple languages. As an example, the modern web application might use 5 or more languages (e.g. SQL, Java, TypeScript, HTML, CSS). Such multilingual codebases present a difficult challenge to the development and maintenance of code analysis tools [1]. To address this challenge current source code analysis tools utilize multiple parsers (one per supported language supported).

Outline:

1. Mention current SIGMA tool.
2. Normalization section is untested.

RG Evaluate and test each step of the normalization proccess used by SIGMA to understand the effectiveness of each step to improve the normalization process of SIGMA.

Organization

II. Background and Related Work

A. Theoretical Foundations

SIGMA

Brief explanation of normalization steps ## Related Studies {#sec:related}

B. Research Contributions

III. Approach

A. Normalization

Next, the trivially merged grammar is normalized to a unique normal form wherein every production is one of the following two forms:

$\langle A \rangle$ ::= $\langle B \rangle$ 'a' ...
$\langle B \rangle$ ::= $\langle A \rangle$ | 'b' | ... | ε

Where production $\langle A \rangle$ represents form F_1 and production $\langle B \rangle$ represent form F_2. These forms were selected to ease comparison of similar productions. Note, as per this normalization, neither form may be recursively defined. The rational behind this, is to ensure that regardless of how productions are nested they will normalize to the same form, for example: Given grammar EG_1

$\langle A \rangle$ ::= 'a' $\langle B \rangle$
$\langle B \rangle$ ::= 'b' 'c'

and grammar EG_2

$\langle A \rangle$ ::= $\langle B \rangle$ 'c'
$\langle B \rangle$ ::= 'a' 'b'

both EG_1 and EG_2 normalize to the same grammar, as follows:

$\langle A \rangle$ ::= 'a' 'b' 'c'

Normalization continues, repeatedly, through the following six processes until stabilization: eliminating unused rules, simplifying productions, merging equivalent rules, eliminating unit rules, expanding productions, and collapsing compatible productions.

1) Eliminating Unused Rules: Remove all rules that are not produced, directly or indirectly, from the start rule. This is accomplished by marking all rules enumerated via a depth first search, and then dropping unmarked rules. When applied to G_3 the grammar is transformed into grammar G_4, as depicted in Fig. 1a.

2) Simplifying Productions: To simplify in-memory grammar representations, the process removes ε embedded inside terms. Next, productions containing one term are replaced with that term. When this step is applied to grammar G_4, it is transformed into grammar G_5, as depicted in Fig. 1b

3) Merging Equivalent Rules: Rules that have identical productions are replaced by a single rule. This new rule is given a name derived from the rules that were merged to create it. In the following example of this step, rules a and b are merged into the rule a+b:

$\langle s \rangle$ ::= $\langle a \rangle$ | $\langle b \rangle$
$\langle a \rangle$ ::= 'a' 'b' $\langle a \rangle$
$\langle b \rangle$ ::= 'a' 'b' $\langle a \rangle$

Which after equivalent rules are merged yields:

$\langle s \rangle$ ::= $\langle a{+}b \rangle$

$\langle S\rangle$ ::= <S_1> | <S_2>
$\langle S_1\rangle$ ::= <A> | <B>
$\langle A\rangle$ ::= ‘a’ ε <B> <C_1>
$\langle C_1\rangle$ ::= ‘c’
$\langle B\rangle$ ::= ‘b’ ‘d’
$\langle S_2\rangle$ ::= <C_2> | <D>
$\langle C_2\rangle$ ::= ‘c’
$\langle D\rangle$ ::= ‘a’ ‘d’ (‘e’ | ‘c’)
| (<C_2> | ‘b’)

(a) Grammar G_4.

$\langle S\rangle$ ::= <S_1> | <S_2>
$\langle S_1\rangle$ ::= <A> | <B>
$\langle A\rangle$ ::= ‘a’ <B> <C_1>
$\langle C_1\rangle$ ::= ‘c’
$\langle B\rangle$ ::= ‘b’ ‘d’
$\langle S_2\rangle$ ::= <C_2> | <D>
$\langle C_2\rangle$ ::= ‘c’
$\langle D\rangle$ ::= ‘a’ ‘d’ (‘e’ | ‘c’)
| (<C_2> | ‘b’)

(b) Grammar G_5.

$\langle S\rangle$ ::= <S_1> | <S_2>
$\langle S_1\rangle$ ::= <A> | <B>
$\langle A\rangle$ ::= ‘a’ <B> <C>
$\langle C\rangle$ ::= ‘c’
$\langle B\rangle$ ::= ‘b’ ‘d’
$\langle S_2\rangle$ ::= <C> | <D>
$\langle D\rangle$ ::= ‘a’ ‘d’ (‘e’ | ‘c’)
| (<C> | ‘b’)

(c) Grammar G_6.

$\langle S\rangle$ ::= <S_1> | <S_2>
$\langle S_1\rangle$ ::= <A> | <B>
$\langle A\rangle$ ::= ‘a’ <B> ‘c’
$\langle B\rangle$ ::= ‘b’ ‘d’
$\langle S_2\rangle$ ::= ‘c’ | <D>
$\langle D\rangle$ ::= ‘a’ ‘d’ (‘e’ | ‘c’)
| (‘c’ | ‘b’)

(d) Grammar G_7.

$\langle S\rangle$ ::= <S_1> | <S_2>
$\langle S_1\rangle$ ::= <A> | <B>
$\langle A\rangle$ ::= ‘a’ <B> ‘c’
$\langle B\rangle$ ::= ‘b’ ‘d’
$\langle S_2\rangle$ ::= ‘c’ | <D>
$\langle D\rangle$ ::= <D_1> | <D_3>
$\langle D_1\rangle$ ::= ‘a’ ‘d’ <D_2>
$\langle D_2\rangle$ ::= ‘e’ | ‘c’
$\langle D_3\rangle$ ::= ‘c’ | ‘b’

(e) Grammar G_8.

$\langle S\rangle$ ::= <A> | <B> | ‘c’ |
<D>
$\langle A\rangle$ ::= ‘a’ ‘b’ ‘d’ ‘c’
$\langle B\rangle$ ::= ‘b’ ‘d’
$\langle D\rangle$ ::= <D_1> | ‘c’ | ‘b’
$\langle D_1\rangle$ ::= ‘a’ ‘d’ <D_2>
$\langle D_2\rangle$ ::= ‘e’ | ‘c’

(f) Grammar G_9.

Fig. 1. Transformed grammars produced during the normalization of grammar G_3.

$\langle a{+}b\rangle$ ::= `‘a’ ‘b’` $\langle a{+}b\rangle$

When this step is applied to grammar G_5, it is transformed into grammar G_6, as depicted in Fig. 1c.

4) Eliminating Unit Rules: All non-terminals with productions of one of the following two forms will have their non-terminal symbols replaced by their rules, and their productions eliminated.

$\langle a\rangle$ ::= $\langle b\rangle$
$\langle a\rangle$ ::= `‘a’`

Elimination of productions of the first form, is derived from Chomsky Normal Form (CNF) [X]. Eliminations of productions of the second form, a derivation from CNF, allows the simplification process to simplify rules of the following form:

$\langle a\rangle$ ::= $\langle b\rangle$ `‘a’ ‘b’`
$\langle b\rangle$ ::= ε

When this step is applied to grammar G_6, it is transformed into grammar G_7, as depicted in Fig. 1d.

5) Expanding Productions: Productions that have nested rules have all nested content replaced by with a non-terminal. The new non-terminal defines a production pointing to their content. When this step is applied to grammar G_7, it is transformed into grammar G_8, as depicted in Fig. 1e.

6) Collapsing Compatible Productions: The final step of the normalization process combines productions that are compatible with each other. This ensures that any non-terminal symbols referenced in a rule will not define a duplicate production. The following provides an example:

$\langle A\rangle$ ::= `‘a’` $\langle B\rangle$
$\langle B\rangle$ ::= `‘b’ ‘c’`
$\langle C\rangle$ ::= `‘c’` | $\langle D\rangle$
$\langle D\rangle$ ::= `‘d’` | `‘e’`

would then collapse to form:

$\langle A\rangle$ ::= `‘a’ ‘b’ ‘c’`
$\langle C\rangle$ ::= `‘c’` | `‘d’` | `‘e’`

When this step is applied to grammar G_8, it is transformed into grammar G_9, as depicted in Fig. 1f.

IV. Experimental Design

A. Goals, Hypotheses, and Variables

Goals: - Evaluate each step of normalization

Hypotheses: - Null hypothesis: each step has no effect on the halstead effort for MCC.

Variables: - Size of grammar as chosen in SIGMA - Blocking, - Halstead/MCC - The steps that the normalization goes through. (2ˆ#numSteps)

B. Design

Blocked factorial design

Reasons - Effect from size found in SIGMA - Interactions from size also seen in SIGMA

C. Experimental Units

D. Data Collection Procedures

E. Analysis Procedures

F. Evaluation of Validity

Conclusion Validity:

Internal Validity:

Construct Validity:

External Validity:

V. Results

A. Data Set Reduction

B. Descriptive Statistics

C. Hypothesis Testing

VI. Interpretation

A. Evaluation of Results

B. Limits of the Study

C. Inferences

D. Lessons Learned

VII. Threats to Validity

VIII. Conclusions and Future Work

A. Summary of Findings

B. Relation to Existing Evidence

C. Impact

D. Limitations

Acknowledgements

References

[1] Z. Mushtaq, G. Rasool, and B. Shehzad, "Multilingual Source Code Analysis: A Systematic Literature Review," *IEEE Access*, vol. 5, pp. 11 307–11 336, 2017, bibtex: mushtaqMultilingualSourceCode2017.