3-2015

Beyond Support and Confidence: Exploring Interestingness Measures for Rule-based Specification Mining

Bui Tien Duy LE

Singapore Management University, btdle.2012@phdis.smu.edu.sg

David LO

Singapore Management University, davidlo@smu.edu.sg

DOI: https://doi.org/10.1109/SANER.2015.7081843

Citation

LE, Bui Tien Duy and David LO. Beyond Support and Confidence: Exploring Interestingness Measures for Rule-based Specification Mining. (2015). *2015 IEEE 22nd International Conference on Software Analysis, Evolution and Reengineering (SANER): March 2-6, Montreál: Proceedings*. 331-340. Research Collection School Of Information Systems.

Available at: https://ink.library.smu.edu.sg/sis_research/2862

Published in 2015 IEEE 22nd International Conference on Software Analysis, Evolution and Reengineering (SANER): March 2-6, Montréal: Proceedings. https://doi.org/10.1109/SANER.2015.7081843

Beyond Support and Confidence: Exploring Interestingness Measures for Rule-Based Specification Mining

Tien-Duy B. Le and David Lo
School of Information Systems
Singapore Management University, Singapore
{btdle.2012,davidlo}@smu.edu.sg

Abstract*—Numerous rule-based specification mining approaches have been proposed in the literature. Many of these approaches analyze a set of execution traces to discover interesting usage rules, e.g., whenever `lock()` is invoked, eventually `unlock()` is invoked. These techniques often generate and enumerate a set of candidate rules and compute some interestingness scores. Rules whose interestingness scores are above a certain threshold would then be output. In past studies, two measures, namely *support* and *confidence*, which are well-known measures, are often used to compute these scores. However, aside from these two, many other interestingness measures have been proposed. It is thus unclear if support and confidence are the best interestingness measures for specification mining. In this work, we perform an empirical study that investigates the utility of 38 interestingness measures in recovering correct specifications of classes from Java libraries. We used a ground truth dataset consisting of 683 rules and recorded execution traces that are produced when we run the DaCapo test suite. We apply 38 different interestingness measures to identify correct rules from a pool of candidate rules. Our study highlights that many measures are on par to support and confidence. Some of the measures are even better than support or confidence and at least one of the measures is statistically significantly better than the two measures. We also find that compositions of several measures with support statistically significantly outperform the composition of support and confidence. Our findings highlight the need to look beyond standard support and confidence to find interesting rules.

I. Introduction

Specification mining, a term first coined by Ammons et al., refers to "a machine learning approach to discovering formal specifications of the protocol the code must obey when interacting with an application program interface" [2]. One popular specification-mining family is *rule-based specification mining algorithms* that express a protocol as a set of rules, e.g., "whenever `IoAcquireRemoveLock` is called, `IoReleaseRemoveLock` must eventually be called", "whenever `File.read()` is called, `File.open()` must have been called before", etc. Dwyer et al. refer to these rules as response and precedence patterns, and many of the properties for model checking are instances of these rules [8]. These rules have also been used to characterize many of the protocols that Windows drivers need to adhere [1]. Since these rules specify temporal orderings of events (i.e., method calls), they are often referred to as *temporal rules* [37], [21], [19], [18]. In this work, we focus on rule-based specification mining algorithms that extract rules from a set of execution traces.

Many existing rule-based specification mining algorithms work by traversing a search space of candidate rules and evaluating the likelihood of each rule to be true based on some interestingness measures [37], [29], [19], [21], [33]. Two measures namely support, which measures the number of times a candidate rule is satisfied in the execution traces, and confidence, which measures how likely the post-condition of a rule is followed, when its pre-condition occurred in an execution trace are often used [37], [29], [19], [21]. Support and confidence have also been used in many other software engineering studies not limited to the identification of temporal rules, e.g., [16], [17].

Although support and confidence have often been used as interestingness measures for rule-based specification mining, the fact is the number of false positives (i.e., wrongly inferred specifications) remains high. Thummalapenta et al. report that the number of false positives of the inferred specifications can go as high as 65% [33]. In fact, support and confidence are often not able to differentiate between correct specifications from false positives. For many cases, false positives share the same support and confidence scores as correct specifications. Interestingly, the data mining and machine learning community has proposed many other interestingness measures, aside from support and confidence, to differentiate between interesting domain concepts, expressed as rules, which are supported by a dataset, from spurious rules [12]. Some of these measures, e.g., odds ratio, are better accepted, in various other communities, e.g., in medicine, as compared to support and confidence [5], [23], [24], [31], [14]. It is possible that some of these measures are better than support or confidence in identifying correct rules. Unfortunately, so far, there have not been any studies that investigate which of the measures are better than others for specification mining. Thus, there is a need to discover if some of these interestingness measures also perform better than support and confidence for rule-based specification mining.

In this work, we aim to address the above mentioned opportunity and need, by performing an empirical study on the utility of these measures for rule-based specification mining. We would like to study whether any of these measures can perform as well or even better than the standard support and confidence measures. We plug each of the 38 interestingness measures to a simple rule-based specification mining tool and investigate the effectiveness of each measure. To perform the study, we implement a rule-based specification mining tool, that extracts rules whose pre- and post-conditions are only composed of one event (i.e., method call) (e.g., the three examples that are presented earlier). Our algorithm generalizes past specification mining algorithms, in particular the algorithm by Yang et al. [37], to support different kinds of interestingness measures. Our tool enumerates the search space of all rules whose pre- and post-conditions are composed of a single event and for each evaluate its score based on each of the 38 interestingness measures. Using a measure, we get a ranked list of rules based on their assigned scores. We then evaluate the effectiveness of each of the measures based on its ability to rank correct rules earlier in the list. We use *correct rules at N* (CR@N) to measure effectiveness.

Our empirical study finds that there are 7 measures that are better than confidence, and 11 measures that are better than support. We also perform statistical tests to see if the differences are significant. Our tests find that odds ratio is statistically significantly better than both confidence and support. We also combine support with various interestingness measures and find that the composition of support and confidence are outperformed by other compositions. The composition of support and odds ratio performs the best and statistically significantly outperforms the composition of support and confidence for a number of settings.

The contributions of this work are as follows:

1) Our work is the first work that investigates the effectiveness of various interestingness measures for rule-based specification mining.
2) We highlight that support and confidence are not the best among 38 interestingness measures. There are more effective interesting measures, which improves CR@N scores of support and confidence by up to 23.08% and 18.52%, respectively. We have performed statistical tests and highlight that odds ratio is a statistically significantly better measure as compared to support and confidence.
3) We combine support with various interestingness measures including confidence. We find that confidence is outperformed by odds ratio and leverage when combined with support.

The structure of the remainder of this paper is as follows. In Section II, we briefly introduce background information on rule-based specification mining and existing interestingness measures. In Section III, we describe our empirical study methodology. We describe our experiment results in Section IV. We discuss related work in Section V. We finally conclude and mention future work in Section VI.

II. Background

TABLE I
Interestingness Measures - Part I

ID	Interestingness Measure	Formula
M_1	Support	$P(AB)$
M_2	Confidence/Precision	$P(B \mid A)$
M_3	Coverage	$P(A)$
M_4	Prevalence	$P(B)$
M_5	Recall	$P(A \mid B)$
M_6	Specificity	$P(\neg B \mid \neg A)$
M_7	Accuracy	$P(AB) + P(\neg A \neg B)$
M_8	Lift/Interest	$\frac{P(AB)}{P(A)P(B)}$
M_9	Leverage	$P(B \mid A) - P(A)P(B)$
M_{10}	Added Value/ Change of Support	$P(B \mid A) - P(B)$
M_{11}	Relative Risk	$\frac{P(B\|A)}{P(B\|\neg A)}$
M_{12}	Jaccard	$\frac{P(AB)}{P(A)+P(B)-P(AB)}$
M_{13}	Certainty Factor	$\frac{P(B\|A)-P(B)}{1-P(B)}$
M_{14}	Odds Ratio	$\frac{P(AB)P(\neg A\neg B)}{P(A\neg B)P(\neg BA)}$
M_{15}	Yule's Q	$\frac{P(AB)P(\neg A\neg B)-P(A\neg B)P(\neg AB)}{P(AB)P(\neg A\neg B)+P(A\neg B)P(\neg AB)}$
M_{16}	Yule's Y	$\frac{\sqrt{P(AB)P(\neg A\neg B)}-\sqrt{P(A\neg B)P(\neg AB)}}{\sqrt{P(AB)P(\neg A\neg B)}+\sqrt{P(A\neg B)P(\neg AB)}}$
M_{17}	Klosgen	$\sqrt{P(AB)}\times$ $\max(P(B \mid A) - P(B), P(A \mid B) - P(A))$
M_{18}	Conviction	$\frac{P(A)P(\neg B)}{P(A\neg B)}$

Temporal Rules. A temporal rule has the form of $A \rightarrow B$, where A and B are two series of events. A and B are referred to as *pre-condition* and *post-condition* of the rule, respectively. Temporal rules are used to specify the ordering of events, where in our setting, each event corresponds to a method invocation. For example, whenever a `lock()` is called, eventually an `unlock()` is called (Resource Locking Protocol) [19]. Temporal rules can express various constraints on how an API should be used. However, it is costly to manually extract temporal rules from software programs. Therefore, many temporal rule mining techniques have been proposed, e.g., [37], [19]. These studies often use *Support*, and *Confidence* as measures to filter uninteresting temporal rules. In our paper, we are interested in two-event temporal rules, i.e., rules whose pre-condition and post-condition consist of one event each. We leave three-or-more-event temporal rules for a future work.

Aside from standard temporal rules which are *forward-eventually rules*, we are also interested in *backward-eventually rules*, c.f., [21]. These two classes correspond to the response and precedence patterns of Dwyer et al. [8]. Forward-eventually rules have a form of $A \rightarrow B$, and its meaning is that any occurrence of A has to be eventually followed by an occurrence of B. For example, the rule $openfile \rightarrow closefile$ means that any invocation of $openfile$ must be followed by $closefile$ eventually. On the other hand, backward-eventually rules have form of $B \leftarrow A$, and its meaning is that any

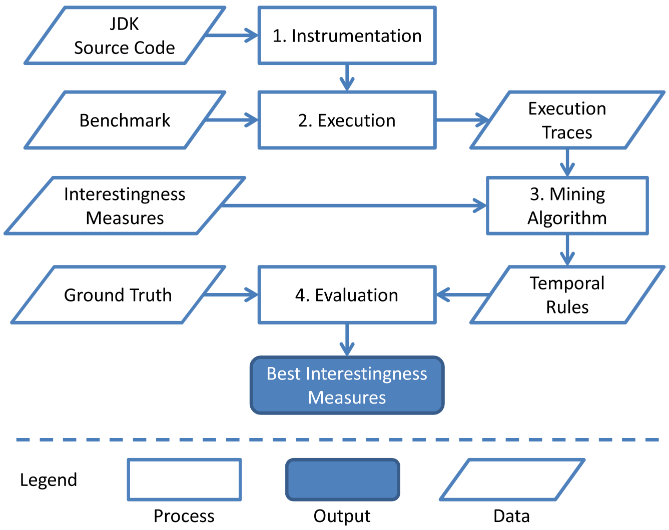

Fig. 1. Study Workflow

TABLE II
INTERESTINGNESS MEASURES - PART II

ID	Interestingness Measure	Formula
M_{19}	Interestingness Weighting Dependency	$(\frac{P(AB)}{P(A)P(B)})^k - 1) \times P(AB)^m$ (We assume $k = 2$ and $m = 2$)
M_{20}	Collective Strength	$\frac{P(AB)+P(\neg B\|\neg A)}{P(A)P(B)+P(\neg A)P(\neg B)} \times \frac{1-P(A)P(B)-P(\neg A)P(\neg B)}{1-P(AB)-P(\neg B\|\neg A)}$
M_{21}	Laplace Correction	$\frac{N(AB)+1}{N(A)+2}$
M_{22}	Gini Index	$P(A) \times (P(B \mid A)^2 + P(\neg B \mid A)^2) + P(\neg A) \times (P(B \mid \neg A)^2 + P(\neg B \mid \neg A)^2) - P(B)^2 - P(\neg B)^2$
M_{23}	Goodman and Kruskal	$\frac{\sum_i \max_j P(A_i B_j) + \sum_j \max_i P(A_i B_j)}{2-\max_i P(A_i)-\max_j P(B_j)} - \frac{\max_i P(A_i)+\max_j P(B_j)}{2-\max_i P(A_i)-\max_j P(B_j)}$
M_{24}	Normalized Mutual Information	$\frac{\sum_i \sum_j P(A_i B_j) \times \log_2 \frac{P(A_i B_j)}{P(A_i)P(B_j)}}{(-\sum_i P(A_i) \log_2 P(A_i))}$
M_{25}	J-Measure	$P(AB) \log \frac{P(B\|A)}{P(B)} + P(A\neg B) \log \frac{P(\neg B\|A)}{P(\neg B)}$
M_{26}	One-Way Support	$P(B \mid A) \log_2 \frac{P(AB)}{P(A)P(B)}$
M_{27}	Two-Way Support	$P(AB) \log_2 \frac{P(AB)}{P(A)P(B)}$
M_{28}	Two-Way Support Variation	$P(AB) \log_2 \frac{P(AB)}{P(A)P(B)} + P(A\neg B) \log_2 \frac{P(A\neg B)}{P(A)P(\neg B)} + P(\neg AB) \log_2 \frac{P(\neg AB)}{P(\neg A)P(B)} + P(\neg A\neg B) \log_2 \frac{P(\neg A\neg B)}{P(\neg A)P(\neg B)}$
M_{29}	ϕ − Coefficient (Linear Correlation Coefficient)	$\frac{P(AB)-P(A)P(B)}{\sqrt{P(A)P(B)P(\neg A)P(\neg B)}}$
M_{30}	Piatetsky-Shapiro	$P(AB) - P(A)P(B)$
M_{31}	Cosine	$\frac{P(AB)}{\sqrt{P(A)P(B)}}$
M_{32}	Loevinger	$1 - \frac{P(A)P(\neg B)}{P(A\neg B)}$
M_{33}	Information Gain	$\log \frac{P(AB)}{P(A)P(B)}$
M_{34}	Sebag-Schoenauer	$\frac{P(AB)}{P(A\neg B)}$
M_{35}	Least Contradiction	$\frac{P(AB)-P(A\neg B)}{P(B)}$
M_{36}	Odd Multiplier	$\frac{P(AB)P(\neg B)}{P(B)P(A\neg B)}$
M_{37}	Example and Counterexample Rate	$1 - \frac{P(A\neg B)}{P(AB)}$
M_{38}	Zhang	$\frac{P(AB)-P(A)P(B)}{\max(P(AB)P(\neg B),P(B)P(A\neg B))}$

occurrence of A has to be preceded by an occurrence of B. For example, the rule $init \leftarrow foo$ means $init$ must be called before foo. For a backward-eventually rule $B \leftarrow A$, A is the pre-condition of the rule and B is the post-condition of the rule. In this work, our mining algorithm extracts both forward-eventually and backward-eventually rules.

Interestingness Measures. Past rule-based specification mining works, e.g., [37], [16], [19], [21], [20], [9], often use or adapt *Support* and *Confidence* measures from the data mining community to differentiate interesting from uninteresting rules. Aside from these two, there are other measures of interestingness that have been proposed in the statistics, machine learning, and data mining communities.

Recently, Geng *et al.* [12] write a comprehensive survey on interestingness measures and they tabulate many measures including *Support* and *Confidence*. Tables I and II show the formulas of 38 measures. These measures are defined based on probabilities and the output of each of these measures corresponds to the interestingness of a rule consisting of two parts A and B, where A is the pre-condition and B is the post-condition of the rule [12]. In Tables I and II, $P(A)$ denotes the probability of the pre-condition A to happen, $P(B)$ denotes the probability of the post-condition B to happen, and so on. The other symbols follow standard probability notations. For example, support is defined as the probability of A and B to happen together (i.e., the proportion of instances where the pre-condition A is followed by the post-condition B). Confidence is defined as the probability of B given A (i.e., the proportion of instances where the post-condition B happens among instances where the pre-condition A happens).

III. Study Approach

Figure 1 shows a workflow of our study. Our workflow takes as input JDK source code, a program benchmark, a list of interestingness measures, and a set of ground truth specifications. The purpose of our study is to find the best interestingness measures for specification mining. To achieve that goal, we first instrument the source code of a number of classes in the Java SDK library, and re-compile the SDK library with the instrumented source code (Step 1). Next, we run the input program benchmark to collect execution traces of instrumented classes (Step 2). These execution traces are then used as input to our mining algorithm (Step 3). The output of the algorithm is a set of temporal rules for each target class, and for each rule, the algorithm computes 38 interestingness scores returned by the interestingness measures listed in Tables I & II. In evaluation step, for each interestingness measure and each target class, we sort the rules based on their interestingness scores, and we compare the resultant ranked list of temporal rules with the ground truth specifications (Step 4). The purpose of this step is to estimate the effectiveness of each interestingness measure using a suitable evaluation metric. In the following sections, we discuss in detail each step in our workflow.

A. Steps 1 & 2: Instrumentation Technique and Execution

Our instrumentation technique inserts new statements to source code of target classes. These statements are placed at entry points of public non-static methods. Every time an instrumented method is invoked, these statements generate an event and log it in a text file which stores the execution trace corresponding to the class containing the method. Each logged event in the text file contains the following information: the method name, the location of the method in the source code, and the identifier of the thread that executes the method.

Table III shows a list of target Java classes and interfaces that we want to instrument. For interfaces and abstract classes, we instrument their sub-classes. After getting the traces, we break the execution traces into smaller traces based on the thread ID of each event. Hence, each of our final execution traces contains method calls from the same thread. We apply our instrumentation technique on the *Java 6* implementation of the target classes. For each class, we collect execution traces by running the 14 Java programs in the DaCapo benchmark [4] (9.12-bach release).[1] These 14 Java programs include popular medium-to-large-size programs such as: Eclipse, ANTLR, HSQLDB, etc.

B. Step 3: Mining Algorithm

We implement a mining algorithm that generalizes past specification mining algorithms, in particular the algorithm by Yang et al. [37], to support different kinds of interestingness measures. Our algorithm utilizes the concept of sliding windows (c.f., [11]) to estimate various probabilities and these probabilities are then used to compute the scores of the 38 interestingness measures shown in Tables I & II. Figure 2 and Figure 3 are the pseudocode for mining forward-eventually rules.

The algorithm in Figure 2 takes as input a program element C which can be an interface, abstract, or concrete class, a set of execution traces of C containing invocations of methods defined in C, and a sliding window size W. The output of our algorithm is a set of forward-eventually temporal rules with 38 interestingness scores computed using the interestingness measures listed in Tables I & II. Lines #1 to #10 generate the set of all possible two-event temporal rules, which are all combinations of two different methods in C. From lines #11 to #25, it processes each of the input traces, one at a time. For each trace, it traverses the trace and maintains a sliding window ($Window$) of size W. At each step, it slides $Window$ one event forward and updates sliding window counts listed in Table IV. We create procedure $UpdateSlidingWindowCounts$ to perform this task (Lines #18 and #23). Next, lines #26 to #35 calculate probability values, which are estimated from the sliding window counts. These probabilities can then be used to compute the 38 interestingness scores. Line #33 stores the scores which are computed using the interestingness measures. Finally, line #36 returns the set of temporal rules along with their interestingness scores.

TABLE III
List of instrumented Java classes & interfaces. The second column stands for the number of instrumented classes corresponding to the class (concrete or abstract) or interface specified in the first column. The third column indicates the number of instrumented methods. The last column specifies the number of ground truth temporal rules for each class or interface.

Class Name	**Instrumented**		**# Ground**
	# Classes	**# Methods**	**Truths**
java.net.DatagramSocket	1	31	58
java.net.MulticastSocket	2	22	30
java.net.Socket	1	43	83
java.net.URL	1	21	15
java.util.ArrayDeque	2	35	32
java.util.ArrayList	2	24	19
java.util.Collection	32	257	11
java.util.Deque	5	135	32
java.util.EnumMap	2	13	12
java.util.EnumSet	4	31	11
java.util.Formatter	1	11	12
java.util.HashMap	1	15	12
java.util.HashSet	3	14	11
java.util.Hashtable	1	18	15
java.util.IdentityHashMap	1	14	12
java.util.LinkedHashMap	2	19	12
java.util.LinkedHashSet	4	17	11
java.util.LinkedList	4	44	38
java.util.List	7	121	35
java.util.Map	13	161	12
java.util.NavigableMap	4	97	33
java.util.NavigableSet	3	69	25
java.util.PriorityQueue	3	20	16
java.util.Queue	15	157	15
java.util.Set	9	78	11
java.util.SortedMap	4	50	18
java.util.SortedSet	4	42	17
java.util.StringTokenizer	1	9	5
java.util.TreeMap	2	38	33
java.util.TreeSet	3	30	25
java.util.WeakHashMap	1	15	12
Total	138	1651	683

TABLE IV
Notations for Sliding Window Count

Notation	**Description**
$N_w(A)$	Number of sliding windows where A exists
$N_w(\neg A)$	Number of sliding windows where A does not exist
$N_w(B)$	Number of sliding windows where B exists
$N_w(\neg B)$	Number of sliding windows where B does not exist
$N_w(AB)$	Number of sliding windows where A is followed by B
$N_w(A\neg B)$	Number of sliding windows where A exists, but B does not exists + Number of sliding windows where both A and B exist, but A is not followed by B
$N_w(\neg AB)$	Number of sliding windows where A does not exist, but B exists
$N_w(\neg A\neg B)$	Number of sliding windows where A does not exist, and B does not exist
N_w	Total number of sliding windows

The procedure in Figure 3 describes how we update the sliding window counts. $UpdateSlidingWindowCounts$ takes as input the set of temporal rules R and the sliding window Q. When the procedure is called, it processes the input set of temporal rules, one at a time, and updates the appropriate sliding window counts. For the sake of brevity, at line #2 it assigns the *pre-condition* event and *post-condition* event of

[1] http://www.dacapobench.org/

```
    Input: $C$: Target interface, abstract or concrete class
           $S$: Set of execution traces of $C$
           $W$: Sliding window size
    Output: Temporal rules with interestingness scores
 1  $R$={} // Set of all possible two-event rules
    // Initialize $R$
 2  foreach public non-static method $m_1$ of $C$ do
 3      foreach public non-static method $m_2$ of $C$ do
 4          if $m_1 \neq m_2$ then
                // $r$ is a new two-event rule
 5              $r$ = new rule
 6              $r.pre \leftarrow m_1$; $r.post \leftarrow m_2$
 7              $R \leftarrow R \cup \{r\}$
 8          end
 9      end
10  end
11  foreach trace $T \in S$ do
12      $Window$={} // sliding window
13      $l \leftarrow \min(W, length(T))$
14      for $i \leftarrow 1$ to $l$ do
15          $T_i$ = $i^{th}$ event in T
16          Append event $T_i$ to the end of $Window$ // enqueue
17      end
        // Update sliding window counts
18      UpdateSlidingWindowCounts($R, Window$)
19      for $i \leftarrow l + 1$ to $length(T)$ do
20          Remove event at the beginning of $Window$ // dequeue
21          $T_i$ = $i^{th}$ event in T
22          Append event $T_i$ to the end of $Window$ // enqueue
            // Update sliding window counts
23          UpdateSlidingWindowCounts($R, Window$)
24      end
25  end
26  foreach rule $r \in R$ do
27      $A \leftarrow r.pre; B \leftarrow r.post$
28      $r.P(A) \leftarrow \frac{r.N_w(A)}{r.N_w}; r.P(\neg A) \leftarrow \frac{r.N_w(\neg A)}{r.N_w}$
29      $r.P(B) \leftarrow \frac{r.N_w(B)}{r.N_w}; r.P(\neg B) \leftarrow \frac{r.N_w(\neg B)}{r.N_w}$
30      $r.P(AB) \leftarrow \frac{r.N_w(AB)}{r.N_w}; r.P(\neg AB) \leftarrow \frac{r.N_w(\neg AB)}{r.N_w}$
31      $r.P(A\neg B) \leftarrow \frac{r.N_w(A\neg B)}{r.N_w}; r.P(\neg A\neg B) \leftarrow \frac{r.N_w(\neg A\neg B)}{r.N_w}$
32      for $i \leftarrow 1$ to 38 do
            // $r.m_i$ stores an interestingness score
               computed using $M_i$ in Tables I & II
33          $r.m_i \leftarrow M_i(r)$
34      end
35  end
36  return $R$
```

Fig. 2. Mining Algorithm

```
    Input: $R$: Set of temporal rules
           $Q$: Sliding window
 1  foreach rule $r \in R$ do
 2      $A \leftarrow r.pre; B \leftarrow r.post$
 3      if $A \in Q$ then
 4          $r.N_w(A) \leftarrow r.N_w(A) + 1$ // Update $N_w(A)$
 5      else
 6          $r.N_w(\neg A) \leftarrow r.N_w(\neg A) + 1$ // Update $N_w(\neg A)$
 7      end
 8      if $B \in Q$ then
 9          $r.N_w(B) \leftarrow r.N_w(B) + 1$ // Update $N_w(B)$
10      else
11          $r.N_w(\neg B) \leftarrow r.N_w(\neg B) + 1$ // Update $N_w(\neg B)$
12      end
13      if $A \notin Q$ and $B \notin Q$ then
            // $A$ and $B$ do not exist in $Q$
14          $r.N_w(\neg A\neg B) \leftarrow r.N_w(\neg A\neg B) + 1$
15      else if $A \notin Q$ and $B \in Q$ then
            // $A$ does not exist, but $B$ exists in $Q$
16          $r.N_w(\neg AB) \leftarrow r.N_w(\neg AB) + 1$
17      else if $A \in Q$ and $B \notin Q$ then
            // $A$ exists, but $B$ does exist in $Q$
18          $r.N_w(A\neg B) \leftarrow r.N_w(A\neg B) + 1$
19      else if $\exists 1 \leq i < j \leq W : Q[i] = A \wedge Q[j] = B$ then
            // $A$ is followed by $B$ in $Q$
20          $r.N_w(AB) \leftarrow r.N_w(AB) + 1$
21      else
            // $A$ is not followed by $B$ in $Q$
22          $r.N_w(A\neg B) \leftarrow r.N_w(A\neg B) + 1$
23      end
24      $r.N_w \leftarrow r.N_w + 1$
25  end
```

Fig. 3. Procedure *UpdateSlidingWindowCounts*

rule r to variable A and B, respectively. From lines #3 to #7, it checks if the *pre-condition* event exists in the sliding window Q, and increases the appropriate count by one (Lines #4 and #6). Similarly, lines #8 to #12 perform the same operations for *post-condition* event. Lines #13 to #23 take into account five cases to update four counts. These counts are $N_w(AB)$, $N_w(\neg AB)$, $N_w(A\neg B)$, and $N_w(\neg A\neg B)$ in Table IV. We consider two cases for $N_w(A\neg B)$ that is when *pre-condition* event exists, but *post-condition* event does not exist, and *pre-condition* event is not followed by *post-condition* event. Finally, we increase the value of N_w by one at line #24, which stores the total number of sliding windows which have been processed so far.

Our mining algorithm is also capable of mining backward-eventually rules. To mine backward-eventually rules, we just have to reverse the order of events in each input execution trace before applying our algorithm, c.f. [18]. For each target class or interface, we apply the mining algorithm twice to mine forward-eventually and backward-eventually rules. Then, we combine the two sets of rules into one for each target class or interface. These combined sets are then input to the evaluation step (step 4).

C. Step 4: Evaluation

In this sub-section, we first describe how we get the ground truth specifications. Next, we describe the metric that we use to evaluate the effectiveness of the 38 interestingness measures for rule-based specification mining.

1) Ground Truth Specifications: Our ground truth specifications are the set of two-event temporal rules including forward-eventually and backward-eventually rules (see Section II). Pradel et al. have formally documented specifications of 32 classes and interfaces in the Java library [27].[2] However, these specifications are in the format of finite state machines where the transitions are labeled with method names. Thus, we need to extract temporal rules from these state machines.

To find ground truth rules, we generate all possible 2-event rules, enumerate each of them, and check if each rule is satisfied by the corresponding Pradel et al.'s finite state machine. To perform these checks, we use the model checker SPIN [15][3]. If the model checker returns "yes" for a rule, that rule is added to the set of ground truth rules. We find that one of the 32 classes whose specification is manually documented by Pradel et al. satisfies no temporal rules – i.e.,

[2] http://mp.binaervarianz.de/icsm2010/index.html

[3] http://spinroot.com/spin/whatispin.html

there is no temporal constraint that governs the ordering in which the methods defined in the class need to be called. We have manually checked this case and find that it is indeed the case that the methods in the class can be used without any constraints. We omit this class and thus end up with 31 classes.

Ground truth rules are of the following formats: $\langle a \rangle \rightarrow \langle b \rangle$ (*forward-eventually* rules) or $\langle a \rangle \leftarrow \langle b \rangle$ (*backward-eventually* rules), where a and b correspond to method calls. For example, *java.net.Socket.connect* → *java.net.Socket.close* is *forward-eventually* rule, which means that when *java.net.Socket.connect* is called, eventually *java.net.Socket.close* must be called. Another example is *java.util.List.*$\langle$*init*$\rangle$ ← *java.util.List.add*, which is a *backward-eventually* rule. This rule means that the constructor of *java.util.List* has to be called before the invocation of *java.util.List.add*. Overall, we extract 683 ground truth temporal rules from 31 Java SDK library's classes and interfaces (see Table III, last column).

2) Evaluation Metric: We use the interestingness measure scores to rank the candidate rules. A more effective interestingness measure will rank correct rules (i.e., rules that are included in the ground truth) higher in the ranked list of rules. To find the most effective interestingness measures, we use *correct rules at N* (**CR@**N), which is defined as the number of correct rules that are identified among the first N sorted candidate rules returned by an interesting measure, as an evaluation metric. In other software engineering studies (e.g., bug triaging and bug localization), this metric has also been referred to as accuracy@N (e.g., [32]) and Hit@N (e.g., [34]).

IV. Experiments & Analysis

In this section, we first describe the set of research questions that we are interested in. Next we describe our experiment results which answer these research questions.

A. Research Questions

To evaluate the interestingness measures, we analyze the following research questions:

Research Question 1 How effective are the interestingness measures for rule-based specification mining? There are many interestingness measures proposed in the literature, however only two of them have been used for specification mining. Are the other 36 measures equally effective for specification mining? To answer this question, we measure the CR@N scores of the 38 measures.

To get the CR@N scores, we follow the workflow presented in Section III. After collecting execution traces, we apply the mining algorithm presented in Section III-B with sliding window size (i.e., W) set to 5, for each of the 31 classes. The output is a set of rules for each class with their interestingness scores. For each of the 31 classes, we then compute CR@N (N=35) scores following the details presented in Section III-C. We take the summation of these CR@N scores, denoted as $\mathbf{\Sigma}$**CR@**N, as proxies of an interestingness measure's effectiveness.

Research Question 2 Which interestingness measures are, if any, *significantly* better than others?

In the previous research question, we have computed CR@N scores as proxies of the measures' effectiveness. Some measures have higher CR@N scores than others. However, are the differences significant? Can any of the 36 interestingness measures *significantly* outperform support and confidence? To answer this research question we perform a series of statistical tests.

To check whether CR@N scores of one measure is significantly different than CR@N scores of another measure, we perform Wilcoxon signed-rank test, which is a well-known non-parametric statistical test [36]. Since we perform Wilcoxon signed-rank test multiple times (i.e., up to 703 times), to avoid getting a statistically significant result by chance (i.e., multiple hypothesis problem), we utilize an alpha correction approach. In particular, we perform Benjamini–Hochberg procedure [3] to correct the multiple hypothesis testing problem. Benjamini–Hochberg procedure is "a more powerful" [22] and more recently proposed method than the Bonferroni correction method [7], to deal with multiple hypothesis testing problem. We utilize the implementation of Wilcoxon signed-rank test in *R* statistical package (version 3.0.2)[4] and the implementation of Benjamini–Hochberg procedure in the *multtest* package (version 2.22) of Bioconductor project[5]. For this research question, we use the same sliding window size of 5, and top-N first candidate rules of 35 (i.e., W=5, N=35) where we discover statistical evidences of the effectiveness of *Support* and *Confidence* compared to other interestingness measures.

Research Question 3 What is the effect of varying the value of N in CR@N formula on the effectiveness of the interestingness measures?

In previous research question, we evaluate the effectiveness of interesting measures by considering number of correct rules in the first 35 candidate rules. In this research question, we inspect the value of $N \in \{5, 10, \ldots, 100\}$. For each N, we calculate the ΣCR@N score corresponding to each interestingness measure, and estimate the rankings of the 38 measures based on their ΣCR@N scores. Measures with higher ΣCR@N scores are assigned lower ranks (i.e., measure with the highest ΣCR@N score is ranked first), and measures with identical ΣCR@N scores are assigned to the same rank. Subsequently, for each measure we compute its average rank over the set of 20 values of N (i.e., $N \in \{5, 10, \ldots, 100\}$). Interestingness measures are then sorted in descending order of their average ranks. We analyze the sorted list of interestingness measures to inspect the effect of N in CR@N formula on their effectiveness.

Research Question 4 What is the effect of combining an interestingness measure with another interestingness measure?

[4] http://www.r-project.org/

[5] http://www.bioconductor.org/packages/release/bioc/html/multtest.html

TABLE V

LIST OF INTERESTINGNESS MEASURES SORTED IN DESCENDING ORDER OF THEIR ΣCR@35 SCORES. "R" = RANKS.

R	ID	Interestingness Measure	ΣCR@35
1	M_{14}	Odds Ratio	96
2	M_9	Leverage	87
3	M_{22}	Gini Index	85
4	M_{13}	Certainty Factor	84
	M_{18}	Conviction	84
	M_{25}	J-Measure	84
7	M_{34}	Sebag-Schoenauer	82
8	M_{10}	Added Value/Change of Support	81
	M_2	Confidence/Precision	81
	M_{37}	Example and Counterexample Rate	81
	M_{28}	Two-Way Support Variation	81
12	M_{21}	Laplace Correction	78
	M_{26}	One-Way Support	78
	M_1	Support	78
15	M_{12}	Jaccard	76
16	M_{35}	Least Contradiction	75
17	M_{24}	Normalized Mutual Information	73
18	M_{31}	Cosine	69
19	M_{17}	Klosgen	68
	M_{32}	Loevinger	68
21	M_{27}	Two-Way Support	65
22	M_{30}	Piatetsky-Shapiro	63
23	M_{36}	Odd Multiplier	62
	M_{29}	$\phi -$ Coefficient (Linear Correlation Coefficient)	62
25	M_{38}	Zhang	61
26	M_{19}	Interestingness Weighting Dependency	59
27	M_{33}	Information Gain	53
	M_8	Lift/Interest	53
29	M_5	Recall	50
30	M_{15}	Yule's Q	46
	M_{16}	Yule's Y	46
32	M_3	Coverage	45
33	M_{11}	Relative Risk	42
34	M_4	Prevalence	35
35	M_{20}	Collective Strength	25
36	M_7	Accuracy	24
37	M_6	Specificity	21
38	M_{23}	Goodman and Kruskal	20

Previous research questions evaluate the effectiveness of each interestingness measure in isolation. However, previous studies in temporal rule mining, e.g., [19], often combine *Support* and *Confidence* together to evaluate candidate temporal rules. In this research question, we inspect the case when *Support* is combined with another interestingness measure in addition to *Confidence*. We select the best interestingness measure identified in research question 3 and combine it with *Support*. To combine an interestingness measure *m* with *Support*, we first select top-N_1 candidate rules returned by *Support*. Then, we re-rank these N_1 rules based on scores calculated by the interestingness measure m. Finally, we compute CR@N_2 of the newly created ranked list to evaluate the effectiveness of the combination between *Support* and *m*. We consider $N_1 \in \{100, 150, 200\}$ which are large enough to capture most of the correct rules, and $N_2 \in \{10, 20, 30, 40, 50\}$.

B. Empirical Analysis

In this subsection, we present our experiment results which answer the proposed research questions.

1) RQ1: Effectiveness of the Measures: Table V shows the ΣCR@35 scores of the 38 interestingness measures. In the table, we sort the measures based on their ΣCR@35. If two measures share the same ΣCR@35 score, they are assigned the same rank. From the table, we find that *Odds Ratio* (M_{14}) outperforms the other measures and it achieves the highest ΣCR@35 score of 96. The second highest ΣCR@35 score is achieved by *Leverage* (i.e., 87). *Goodman and Kruskal* has the lowest ΣCR@35 score (i.e., 20).

Most importantly, we find that the two widely used interestingness measures in many specification mining works (i.e. *Support* and *Confidence*) are ranked 12^{th} (ΣCR@35=78) and 8^{th} (ΣCR@35=81) in Table V, respectively. Comparing the ΣCR@35 scores, *Odds Ratio* outperforms *Support* and *Confidence* by 23.08% and 18.52%, respectively. Aside from *Confidence* and *Odds Ratio*, 9 other measures outperform *Support* by 3.85% to 11.54%. Aside from *Odds Ratio*, 6 other measures outperform *Confidence* by 1.23% to 7.41%. These results indicate that other interestingness measures, in addition to *Confidence* and *Support*, can be utilized for specification mining.

Odds Ratio and Leverage are the two best interestingness measures and both of them outperform Support and Confidence. Support and Confidence are biased towards events that appear many times. If two events are unrelated but appear many times, it is highly likely that these events randomly occur in a window in a particular order many times. This will result in the pair of events to achieve high Support and Confidence scores albeit they are unrelated. Odds Ratio deals with this weakness, by considering not only then number of times the two events happens together (i.e., $P(AB)$) but also the number of times an event happens and the other does not (i.e., $P(A\neg B)$) and the number of times both events do not happen together (i.e., $P(\neg A\neg B)$). Leverage deals with this weakness, by subtracting confidence (i.e., $P(B \mid A)$) with the likelihood of two events to co-occur together by chance (i.e., $P(A)P(B)$).

2) RQ2: Significantly Better Measures: To answer this research question, we perform a Wilcoxon signed-rank test at a significance level of 0.05 for every interestingness measure pair. The input to the Wilcoxon signed-rank test is two vectors of CR@35 scores corresponding to the two measures in the pair. Since we have 38 interestingness measures, we have to run Wilcoxon signed-rank test $\frac{38\times37}{2} = 703$ times. Nevertheless, conducting multiple hypothesis tests might trigger the multiple comparison problem [30], [6]. Therefore, we apply Benjamini–Hochberg procedure to adjust p-values output by multiple Wilcoxon signed-rank tests [3]. For any pair, if the adjusted *p-value* is less than 0.05, we conclude that the two measures are significantly different in terms of their CR@35 scores. In total, we are able to identify 378 pairs whose members are significantly different from each other. We refer to such pairs as *significant pairs*.

To represent the results of our statistical tests compactly, we create a partial order where each node is a measure and each edge from node M_i to node M_j denotes that measure M_i statistically significantly outperforms M_j (i.e., the p-value output by Wilcoxon signed-rank test and adjusted by Benjamini–Hochberg procedure is less than 0.05 as well as the CR@35 score of M_i is greater than that of M_j). To simplify the graph, we do not draw edges that can be inferred by transitivity (i.e., we omit an edge from M_i to M_k if there is an edge from M_i to M_j and there is another edge from M_j

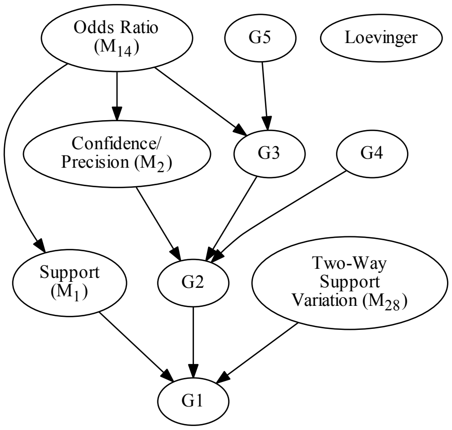
Odds Ratio (M$_{14}$)
G5
Loevinger
Confidence/ Precision (M$_2$)
G3
G4
Support (M$_1$)
G2
Two-Way Support Variation (M$_{28}$)
G1

Fig. 4. Partial order depicting the relationships among the top-30 interesting measures. G1, G2, G3, G4, and G5 are groups of interestingness measures. Table VI lists out members of each group.

TABLE VI
MEMBERS OF INTERESTINGNESS MEASURE GROUPS IN FIGURE 4

Group	Members
G1	Information Gain (M_{33}), Interestingness Weighting Dependency (M_{19}), Lift/Interest (M_8), Recall (M_5), Yule's Q (M_{15}), Yule's Y (M_{16})
G2	Cosine (M_{31}), Klosgen (M_{17}), Laplace Correction (M_{21}), Least Contradiction (M_{35}), Odd Multiplier (M_{36}), $\phi -$Coefficient (Linear Correlation Coefficient) (M_{29}), Piatetsky-Shapiro (M_{30}), Two-Way Support (M_{27}), Zhang (M_{38})
G3	Example and Counterexample Rate (M_{37}), Jaccard (M_{12}), Normalized Mutual Information (M_{24}), Sebag-Schoenauer (M_{34})
G4	Added Value/Change of Support (M_{10}), Gini Index (M_{22}), J-Measure (M_{25}), Leverage (M_9), One-Way Support (M_{26})
G5	Certainty Factor (M_{13}), Conviction (M_{18})

to M_k.).

Figure 4 shows the partial order depicting the relationships among the top-30 interestingness measures. In the graph, *Odds Ratio* is one of a few measures whose in-degrees are zeros. Among the represented measures, *Odds Ratio*, which has the highest ΣCR@35 score, has significantly higher ΣCR@35 scores than 21 out of the top-30 measures including *Confidence* and *Support*. This implies *Odds Ratio* is a good measure for specification mining.

Measures in the equivalence classes share the same set of measures that they statistically significantly outperform and the same set of measures that statistically significantly outperform them. The fact that some formulas are in the same equivalence class may indicate semantic similarities among them, especially if their CR@N scores are very similar. To check this hypothesis, we look at similarities among formulas, and find that formulas that are very similar to one another (e.g., Yule's Q and Yule's Y, and Lift/Interest and Information Gain) are in the same class and have very similar CR@N scores. However, it is not necessarily the case that all pairs of formulas in the same equivalence class are semantically similar. Since they can be in the same class simply because they perform almost equally well for the specification mining tasks investigated in this work.

TABLE VII
LIST OF INTERESTINGNESS MEASURES SORTED IN ASCENDING ORDER OF THEIR AVERAGE RANKS.

R	ID	Interestingness Measure	Average Rank
1	M_{14}	Odds Ratio	3.20
2	M_9	Leverage	5.25
3	M_2	Confidence/Precision	5.35
4	M_{37}	Example and Counterexample Rate	5.50
	M_{34}	Sebag-Schoenauer	5.50
6	M_{21}	Laplace Correction	6.85
7	M_{22}	Gini Index	8.90
8	M_1	Support	9.25
9	M_{12}	Jaccard	12.60
10	M_{25}	J-Measure	12.95
11	M_{35}	Least Contradiction	13.10
12	M_{28}	Two-Way Support Variation	13.25
13	M_{31}	Cosine	14.30
14	M_{32}	Loevinger	15.75
15	M_{13}	Certainty Factor	16.10
	M_{18}	Conviction	16.10
17	M_4	Prevalence	16.15
18	M_{26}	One-Way Support	17.00
19	M_{10}	Added Value/Change of Support	17.25
20	M_{24}	Normalized Mutual Information	17.50
21	M_{27}	Two-Way Support	18.85
22	M_{36}	Odd Multiplier	20.90
23	M_{38}	Zhang	21.00
24	M_{30}	Piatetsky-Shapiro	21.65
25	M_{33}	Information Gain	22.90
	M_8	Lift/Interest	22.90
27	M_{17}	Klosgen	23.10
28	M_{19}	Interestingness Weighting Dependency	23.80
29	M_{29}	$\phi -$ Coefficient (Linear Correlation Coefficient)	24.50
30	M_{15}	Yule's Q	27.30
	M_{16}	Yule's Y	27.30
32	M_{11}	Relative Risk	29.55
33	M_5	Recall	29.70
34	M_3	Coverage	32.35
35	M_{20}	Collective Strength	34.75
36	M_7	Accuracy	35.30
37	M_{23}	Goodman and Kruskal	35.95
38	M_6	Specificity	37.00

3) RQ3: Effect of Cut-Point N: In this research question, we inspect the effect of cut-point N by varying its value from 5 to 100 with a step of 5 (i.e., $N \in \{5, 10, \ldots, 100\}$). We sort the interestingness measures based on their average ranks over the set of 20 values of N. Table VII shows the list of interestingness measures in ascending order of their average ranks. From the table, *Odds Ratio* has the best average rank of 3.20. *Confidence* (M_2) achieves the third best average rank of 5.25, whereas *Support* (M_1) is at position #8 (average rank of 9.25). Aside from *Odds Ratio*, *Leverage* also outperforms *Confidence*. There are 5 other interestingness measures aside from *Odds Ratio* and *Confidence* that outperform *Support*. Furthermore, using Wilcoxon signed-ranked test and Benjamin–Hochberg procedure (at significance level of 0.05), we discover that *Odds Ratio* and *Leverage* statistically significantly outperform *Support*.

4) RQ4: Effectiveness of Combined Measures: Table VIII shows the evaluation results of comparing the combination of M_1 (*Support*) and M_2 (*Confidence*), and the combinations of M_1 (*Support*) and M_i where $M_i \in \{$ M_{14} (*Odds Ratio*), M_9 (*Leverage*), M_{37} (*Example and Counterexample Rate*), M_{34} (*Sebag-Schoenauer*)$\}$. These four measures achieve the best average ranks shown in Table VII. From Table VIII, we find that *Support* and *Odds Ratio* together can identify more correct rules than *Support* and *Confidence* in most cases

TABLE VIII
ΣCR@N_2 SCORES OF THE COMBINATION BETWEEN *Support* (M_1) AND OTHER INTERESTINGNESS MEASURES INCLUDING *Confidence* (M_2) AND *Odds Ratio* (M_{14}). “*” INDICATES THAT THE DIFFERENCE BETWEEN *Confidence*, AND THE CORRESPONDING INTERESTINGNESS MEASURE, WHEN COMBINED WITH *Support* IS STATISTICALLY SIGNIFICANT (SIGNIFICANCE LEVEL = 0.05).

N_1	N_2	*Support* (M_1) combines with				
		M_2	M_{14}	M_9	M_{37}	M_{34}
100	10	20	31	26	20	20
100	20	45	56	46	43	43
100	30	63	82	68	63	63
100	40	92	104	91	92	92
100	50	108	114	106	108	108
150	10	25	37	29	25	25
150	20	49	62	53	47	47
150	30	71	81	71	71	71
150	40	95	95	96	95	96
150	50	109	118	112	109	109
200	10	24	37*	29	24	24
200	20	49	62	52	47	47
200	30	69	82*	73	69	69
200	40	93	100	92	93	94
200	50	107	114	114	107	107

(i.e., 14 out of 15 cases). For the remaining one case (i.e., $N_1 = 150 \wedge N_2 = 40$), they identify the same number of correct rules. From the table, the combination of *Example & Counterexample Rate* (M_{37}), and *Sebag-Schoenauer* (M_{34}) with *Support* are comparable where their ΣCR@N scores are identical in 13 out of 15 cases. We also perform Wilcoxon signed rank test (at significance level of 0.05) and apply Benjamini–Hochberg procedure to adjust p-values for multiple hypothesis testing. We discover that *Support* and *Odds Ratio* statistically significantly outperforms *Support* and *Confidence* for the following settings: ($N_1 = 200 \wedge N_2 = 10$), ($N_1 = 200 \wedge N_2 = 30$).

C. Threats to Validity

We have a few threats to validity that we need to acknowledge. Threats to internal validity relate to errors in our experiments. We have rechecked our experiments. Still there could be errors that we did not notice. Threats to external validity relate to the generalizability of our findings. We have investigated 38 different measures and used 683 ground truth specifications to evaluate them. We have also mined specifications from the execution traces of 14 medium-large projects from the DaCapo benchmark that capture how these 14 projects make use of java.util and java.net API classes. Although the size of these API classes are small, the size of the client applications that we run is rather large. The client applications, which include Eclipse, Jython, Lucene, Tomcat, and Xalan, are all medium-large projects. Admittedly, we have only investigated the benefit of the various effectiveness measures on inferring specifications for 31 Java SDK classes using one specification mining algorithm. Also, admittedly, all the 14 programs that we investigate in this work are written in Java. In the future, we plan to reduce these threats further by considering more ground truth specifications from more APIs, more traces collected from more programs written in various programming languages, and additional specification mining algorithms. Threats to construct validity refer to the suitability of our evaluation metric. We have used correct rules at N (CR@N), Wilcoxon signed-rank test, and Benjamini–Hochberg procedure. Since a better interestingness measure should assign higher scores to correct than wrong rules, CR@N that measures the number of correct rules in the top-N rules with the highest scores is a suitable evaluation metric. CR@N, which is also referred to as accuracy@N (e.g., [32]) or Hit@N (e.g., [34]), has been used as an evaluation metric in many past software engineering studies that also produce a list of results for developer inspection [32], [28], [34]. Wilcoxon signed-rank test is a standard test proposed by the statistics community and it has also been used in many past software engineering studies [13]. Benjamini–Hochberg procedure is an alpha-correction method used to deal with multiple hypothesis testing problem [3].

V. Related Work

Rule-Based Specification Mining. There is a number of studies that propose various techniques to extract rules from code or execution traces. Li et al. extract rules from source code methods by employing association rule mining technique [16]. They convert a method into a transaction, which is a set of items where an item is a method invocation that appears in the method body. These transactions are then input to association rule mining which uses support and confidence to filter rules that are deemed uninteresting. Yang et al. extract two-event rules from execution traces containing logs of methods that are called when an instrumented program is run [37]. To find these two-event rules they consider the search space of all candidate two-event rules and evaluate the interestingness of each of the rules using the notion of satisfaction rate which is analogous to confidence. They break a long trace into partitions and compute the satisfaction rate of a rule by counting the number of partitions where the rule holds divided by the total number of partitions. Lo et al. extend Yang et al.'s work by mining temporal rules of arbitrary lengths [19]. To make the approach scales, they propose new search space pruning strategies to cut the sub-search spaces containing uninteresting rules en-masse. They also consider the notions of confidence and support to measure the interestingness of rules.

Recently, a number of more advanced techniques have been proposed to mine more complex rules. Lo et al. mine rules enriched with quantification [21]. With quantification, users can mine rules that specify data flow constraints between two method invocations, e.g., the output of one method invocation is the x^{th} input of another method invocation. Lo et al. also mine rules following the semantics of Live Sequence Charts (LSCs) which are enriched with Daikon-style constraints to serve as guards [20]. For all of the above studies, support and confidence have been used as the interestingness measures.

Usages of Interestingness Measures. Many past studies have demonstrated the utility of many of the interestingness measures investigated in this study. Odds ratio is often used in biomedical studies [5], [23]. It is often used to indicate the odds of a particular event to occur (e.g., a health outcome) after a particular medical treatment has been given or an exposure to a particular substance [24], [31], [14]. Relative risk has also been used in biomedical studies. For example, Yusuf et al. have used relative risk to measure the effect of taking a number of supplements (e.g., folic acid, B6, and B12) on the risk of major cardiovascular events [38]. Fawcet and Provost have used certainty factor to identify frauds by analyzing user behavioral changes [10]. Yule's Q and Yule's Y have often been used in social science [35], [25]. Weede uses Yule's Q to investigate the association between democracy and war involvement [35]. Page and Shapiro use Yule's Y to investigate the association between public opinion and policy in the United States [25]. Normalized mutual information has been used in various image processing studies [26]. There are many other studies that have used various interestingness measures for various purposes. Due to page limitation, we do not discuss them here.

VI. Conclusion and Future Work

In this work, we investigate the effectiveness of 38 interestingness measures for rule-based specification mining. Only two of them, namely support and confidence, have been used in past specification mining studies. Our study finds that support and confidence are not the best measures among the 38 interestingness measures; in terms of *correct rules at* N (CR@N) scores, there are other interestingness measures that outperform support and confidence by up to 23.08% and 18.52%, respectively. We also find that odds ratio and leverage outperform confidence when each of them is paired with support. Each of the mined rules can be used as input to a model checker or a lightweight verification tool to find bugs, c.f., [33]. If many rules are false positives, there would be many false positive warnings. On the other hand, more true positive rules can translate to more real bugs being identified. In this paper, we have shown that aside from support and confidence, other interestingness measures like odds ratio can be used to increase the number of true positives and reduce the number of false positives when only the top N rules are used for further analysis.

As future work, we plan to incorporate odds ratio to more advanced rule-based specification mining solutions, e.g., those that can mine rules with guards [20], etc. We also plan to expand our empirical study to include specifications of classes from other libraries and execution traces extracted from additional client programs. Furthermore, each interestingness measure captures different aspects of the interestingness of a rule. Past studies have only used two interestingness measures together (e.g., support and confidence). In a future work, we plan to combine many more measures to form a composite measure that can better differentiate correct from spurious rules (i.e., false positives).

References

[1] "Rules for WDM drivers," http://msdn.microsoft.com/en-us/library/windows/hardware/ff551714(v=vs.85).aspx (Last accessed: 20/01/2014).

[2] G. Ammons, R. Bodík, and J. R. Larus, "Mining specifications," in *POPL*, 2002.

[3] Y. Benjamini and Y. Hochberg, "Controlling the false discovery rate: a practical and powerful approach to multiple testing," *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 289–300, 1995.

[4] S. M. Blackburn, R. Garner, C. Hoffmann, A. M. Khan, K. S. McKinley, R. Bentzur, A. Diwan, D. Feinberg, D. Frampton, S. Z. Guyer, M. Hirzel, A. L. Hosking, M. Jump, H. B. Lee, J. E. B. Moss, A. Phansalkar, D. Stefanovic, T. VanDrunen, D. von Dincklage, and B. Wiedermann, "The dacapo benchmarks: java benchmarking development and analysis," in *OOPSLA*, 2006, pp. 169–190.

[5] J. Cornfield, "A method for estimating comparative rates from clinical data. applications to cancer of the lung, breast, and cervix," *Journal of the National Cancer Institute*, vol. 11, pp. 1269–1275, 1951.

[6] S. Dudoit, J. P. Shaffer, and J. C. Boldrick, "Multiple hypothesis testing in microarray experiments," *Statistical Science*, pp. 71–103, 2003.

[7] O. J. Dunn, "Multiple comparisons among means," *Journal of the American Statistical Association*, vol. 56, no. 293, pp. 52–64, 1961.

[8] M. B. Dwyer, G. S. Avrunin, and J. C. Corbett, "Patterns in property specifications for finite-state verification," in *ICSE*, 1999, pp. 411–420.

[9] D. Fahland, D. Lo, and S. Maoz, "Mining branching-time scenarios," in *ASE*, 2013.

[10] T. Fawcett and F. Provost, "Adaptive fraud detection," *Data mining and knowledge discovery*, vol. 1, no. 3, pp. 291–316, 1997.

[11] M. Gabel and Z. Su, "Online inference and enforcement of temporal properties," in *ICSE*, 2010, pp. 15–24.

[12] L. Geng and H. J. Hamilton, "Interestingness measures for data mining: A survey," *ACM Comput. Surv.*, vol. 38, no. 3, Sep. 2006.

[13] M. Gethers, B. Dit, H. H. Kagdi, and D. Poshyvanyk, "Integrated impact analysis for managing software changes," in *ICSE*, 2012, pp. 430–440.

[14] J. Henning, D. U. Pfeiffer *et al.*, "Risk factors and characteristics of h5n1 highly pathogenic avian influenza (hpai) post-vaccination outbreaks," *Veterinary research*, 2009.

[15] G. Holzmann, *Spin Model Checker, the: Primer and Reference Manual*, 1st ed. Addison-Wesley Professional, 2003.

[16] Z. Li and Y. Zhou, "Pr-miner: automatically extracting implicit programming rules and detecting violations in large software code," in *ESEC/SIGSOFT FSE*, 2005.

[17] V. B. Livshits and T. Zimmermann, "Dynamine: finding common error patterns by mining software revision histories," in *FSE*, 2005.

[18] D. Lo, S. Khoo, and C. Liu, "Mining past-time temporal rules from execution traces," in *WODA*, 2008, pp. 50–56.

[19] D. Lo, S.-C. Khoo, and C. Liu, "Mining temporal rules for software maintenance," *Journal of Software Maintenance*, vol. 20, no. 4, 2008.

[20] D. Lo and S. Maoz, "Scenario-based and value-based specification mining: better together," *Autom. Softw. Eng.*, vol. 19, no. 4, 2012.

[21] D. Lo, G. Ramalingam, V. P. Ranganath, and K. Vaswani, "Mining quantified temporal rules: Formalism, algorithms, and evaluation," *Sci. Comput. Program.*, 2012.

[22] J. H. McDonald, *Handbook of Biological Statistics*, 2009, vol. 2.

[23] M. L. McHugh, "The odds ratio: calculation, usage, and interpretation," *Biochemia Medica*, vol. 19, no. 2, pp. 120–126, 2009.

[24] C. Mutegi, H. Ngugi, S. Hendriks, and R. Jones, "Prevalence and factors associated with aflatoxin contamination of peanuts from western kenya," *International journal of food microbiology*, vol. 130, no. 1, 2009.

[25] B. I. Page and R. Y. Shapiro, "Effects of public opinion on policy," *The American Political Science Review*, pp. 175–190, 1983.

[26] J. P. Pluim, J. A. Maintz, and M. A. Viergever, "Mutual-information-based registration of medical images: a survey," *IEEE Transactions on Medical Imaging*.

[27] M. Pradel, P. Bichsel, and T. R. Gross, "A framework for the evaluation of specification miners based on finite state machines," in *ICSM*, 2010.

[28] S. Rao and A. C. Kak, "Retrieval from software libraries for bug localization: a comparative study of generic and composite text models," in *MSR*, 2011.

[29] H. Safyallah and K. Sartipi, "Dynamic analysis of software systems using execution pattern mining," in *ICPC*, 2006, pp. 84–88.

[30] J. P. Shaffer, "Multiple hypothesis testing," *Annual review of psychology*, 1995.

[31] M. J. Stampfer, "Welding occupations and mortality from parkinson's disease and other neurodegenerative diseases among united states men, 1985–1999," *Journal of occupational and environmental hygiene*, 2009.

[32] A. Tamrawi, T. T. Nguyen, J. M. Al-Kofahi, and T. N. Nguyen, "Fuzzy set and cache-based approach for bug triaging," in *SIGSOFT FSE*, 2011.

[33] S. Thummalapenta and T. Xie, "Mining exception-handling rules as sequence association rules," in *ICSE*, 2009, pp. 496–506.

[34] S. Wang and D. Lo, "Version history, similar report, and structure: Putting them together for improved bug localization," in *ICPC*, 2014.

[35] E. Weede, "Some simple calculations on democracy and war involvement," *Journal of Peace Research*, pp. 377–383, 1992.

[36] F. Wilcoxon, "Individual comparisons by ranking methods," *Biometrics Bulletin*, vol. 1, no. 6, pp. 80–83, 1945.

[37] J. Yang, D. Evans, D. Bhardwaj, T. Bhat, and M. Das, "Perracotta: Mining temporal api rules from imperfect traces," in *ICSE*, May 2006.

[38] S. Yusuf, P. Sleight, J. Pogue, J. Bosch, R. Davies, and G. Dagenais, "Effects of an angiotensin-converting-enzyme inhibitor, ramipril, on cardiovascular events in high-risk patients. the heart outcomes prevention evaluation study investigators." *The New England journal of medicine*, vol. 342, no. 3, pp. 145–153, 2000.