A

The Truth, the Whole Truth, and Nothing but the Truth: A Pragmatic Guide to Assessing Empirical Evaluations

Stephen M. Blackburn, Amer Diwan, Matthias Hauswirth, Peter F. Sweeney,

José Nelson Amaral, Tim Brecht, Lubomr Bulej, Cliff Click, Lieven Eeckhout, Sebastian Fischmeister, Daniel Frampton, Laurie J. Hendren, Michael Hind, Antony L. Hosking, Richard E. Jones, Tomas Kalibera, Nathan Keynes, Nathaniel Nystrom, and Andreas Zeller

An unsound claim can misdirect a field, encouraging the pursuit of unworthy ideas and the abandonment of promising ideas. An inadequate description of a claim can make it difficult to reason about the claim, for example to determine whether the claim is sound. Many practitioners will acknowledge the threat of unsound claims or inadequate descriptions of claims to their field. We believe that this situation is exacerbated and even encouraged by the lack of a systematic approach to exploring, exposing, and addressing the source of unsound claims and poor exposition.

This paper proposes a framework that identifies three sins of reasoning that lead to unsound claims and two sins of exposition that lead to poorly described claims. Sins of exposition obfuscate the objective of determining whether or not a claim is sound, while sins of reasoning lead directly to unsound claims.

Our framework provides practitioners with a principled way of critiquing the integrity of their own work and the work of others. We hope that this will help individuals conduct better science and encourage a cultural shift in our research community to identify and promulgate sound claims.

Categories and Subject Descriptors: C.4 [**Computer Systems Organization**]: Performance of Systems

General Terms: Performance evaluation (efficiency and effectiveness)

Additional Key Words and Phrases: experimental evaluation, observation study, experimentation

1. INTRODUCTION

To understand the behavior of a system or to understand the efficacy of an idea, we conduct evaluations and then derive and form claims that attempt to draw meaning from the evaluations. An evaluation is either an experiment or an observational study, consisting of steps performed and data produced from those steps. A claim is an assertion about the significance and meaning of an evaluation that has already been conducted; thus, unlike a *hypothesis* which precedes an evaluation, a claim comes after the evaluation. The dissemination of claims and evaluations may be formal or informal, and can take many forms, including a talk, a paper, an appendix, or an associated artifact. Evaluation is central to our discipline, because, as Brooks [1996] argues, we are a design discipline, therefore we must test our ideas by their usefulness.

When we derive a claim that the evaluation does not support, our claim is unsound: the claim may incorrectly characterize the behavior of the system or state that an idea is beneficial when it is not (or vice versa). Concretely, a claim is sound if the evaluation (i) provides all the evidence [the whole truth] necessary to support the claim; and (ii) does not provide any evidence [nothing but the truth] that contradicts the claim.

DOI:http://dx.doi.org/10.1145/0000000.0000000

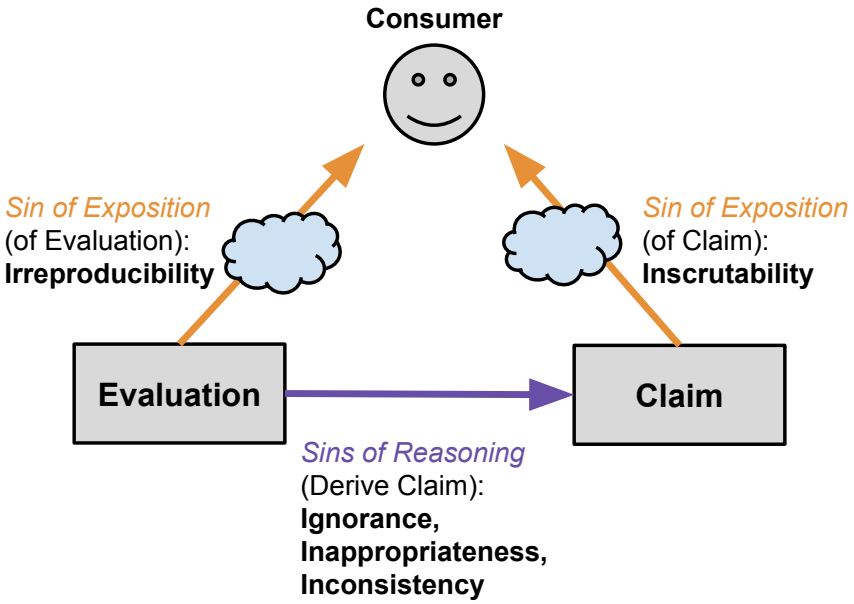

Fig. 1. Sins of exposition and sins of reasoning. The sins of exposition obfuscate the evaluation and/or the claim, preventing an observer from determining the soundness of the claim. The sins of reasoning interfere with the derivation of the claim, resulting in an unsound claim.

Furthermore, we demonstrate in Section 1.1 that a claim is unsound if it suffers from one or more sins of reasoning. In contrast, an evaluation is what it is: it is neither sound nor unsound. Rather, an evaluation may be uninteresting or too weak to derive useful claims.

For the progress of a field, it is not enough for the claim to be merely sound: imagine a situation where experimentalists put out claims but do not describe in detail how they arrived at the claim. Without such details the community at large has no way of determining which claims are sound and which are unsound: thus, no one knows which ideas are worth building upon and which are not. Consequently the field progresses only by accident. When either the claim or the evaluation is inadequately described, the description suffers from one or more sins of exposition.

Figure 1 shows the relationship between the Claim and the Evaluation and where sins of exposition and sins of reasoning arise. The remainder of this paper systematically explores these sins.

1.1. Sins of reasoning

Figure 2 shows all the possible relationships between the scope of a claim and the scope of its supporting evaluation. The scope of a claim is the set of facts (data, methodology, etc.) that the claim assumes. The scope of an evaluation is the set of facts that describe all aspects of the evaluation including any data that the evaluation produces. A claim is sound when the scope of the claim and scope of the evaluation perfectly overlap. When they do not overlap, anything additional in the claim's scope must be either general knowledge (e.g., the laws of physics) or cited (e.g., some result in the literature). Given these two scopes, one or more of (a), (b), or (c) may be empty: thus, the full space has 8 combinations, some of which produce sound claims while others may produce unsound claims. Because we assume that there is a claim and there is an evaluation,

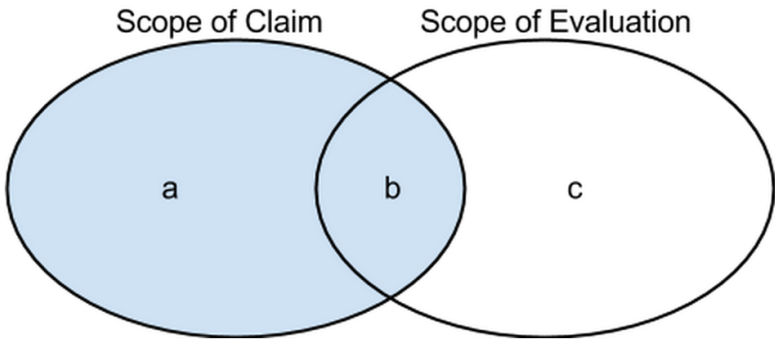

Fig. 2. Relationship between the scopes of the claim and the evaluation.

it must be the case that: (a) $\cup$ (b) $\neq \phi$ and (b) $\cup$ (c) $\neq \phi$, where $\cup$ is set union and ϕ represents the empty set. This assumption eliminates three ((a) and (b) are both empty, (b) and (c) are both empty, or (a) and (b) and (c) are all empty) of the eight possible combinations. In addition, when only (b) $\neq \phi$, the claim and the evaluation perfectly overlap; consequently, our claim is sound. This leaves us with four remaining possibilities, each of which may lead to an unsound claim:

(1) (a) is empty, (b) and (c) are not empty. Thus, the evaluation provides support for the claim (i.e., (b)) but includes other facts from the evaluation (i.e., (c)) that the claim ignores. If (c) does not invalidate the claim then some aspects of the evaluation may have been wasteful: e.g., the evaluation collected data that was never used in the claim. On the other hand, if (c) invalidates the claim, then the claim is unsound. We say such a claim commits the *sin of ignorance* because it ignores important facts from the evaluation.
(2) (c) is empty, (a) and (b) are not empty. Thus, the claim extends beyond the evaluation; in other words, the claim relies on aspects that are actually missing from the evaluation. If (a) relies on anything other than established knowledge (e.g., laws of physics or cited literature), we say such a claim commits the *sin of inappropriateness*.
(3) (b) is empty, (a) and (c) are not empty. Thus the claim and evaluation are disjoint: the claim and evaluation are essentially about different things. We can think of this as an extreme case of the sin of ignorance and the sin of inappropriateness but there is one special case here that we highlight because we have encountered it frequently. Specifically, we mean the situation where we compare two incompatible entities in the evaluation and the claim ignores this incompatibility. We say such a claim commits the *sin of inconsistency* because it is based on an evaluation that compares two entities that are not comparable (i.e., apples and oranges).
(4) None of (a), (b), or (c) are empty. This is again a combination of the *sin of ignorance* and the *sin of inappropriateness*. When evaluating the relationship between the scopes of a claim and an evaluation, we should not assume that all of the sins cleanly fall into just one category.

It is our intention that the framework in this paper will lead to identifying new sources of sins and improve community standards. As we show in this paper, these sins are non-obvious: even when following the best community practices we may commit these sins.

1.2. Sins of exposition

Having a claim that perfectly matches the underlying evaluation is not enough: we have to adequately communicate the claim and the evaluation in clear enough terms that others can properly interpret and assess our work.

The sin of inscrutability occurs when the description of the claim is inadequate. Authors may neglect to explicitly identify a claim at all; they may make claims that are ambiguous; or they may express their claim in a way that unintentionally distorts their intent. Because a claim is synthesized by its author, it should always be possible to avoid the sin of inscrutability.

The sin of irreproducibility occurs when the description of the evaluation is inadequate. This sin is the failure of the authors to clearly communicate an evaluation, including the steps that were taken in the course of the evaluation and the data that it produced. In addition to hampering the reproduction of the evaluation, irreproducibility also makes the evaluation unclear, clouding its relationship to the claim. In contrast to a claim, an evaluation is a natural process; i.e., an unbounded number of factors in the environment may affect the outcome of the evaluation in subtle ways. While the relevance of some variables to an evaluation (such as the model of computer or set of benchmarks used) is obvious, the relevance of other variables is much less obvious (e.g., a news event leads to a changed load seen by the search engine being evaluated). For this reason, irreproducibility is hard to avoid entirely.

The sins of exposition interact with sins of reasoning: if an evaluation is irreproducible or a claim is inscrutable, others have no way of establishing whether or not the claim suffers from any sins of reasoning.

2. CONTRIBUTIONS

Sound claims are difficult to produce: a subtle incompatibility between the scope of the claim and the scope of the evaluation can render a claim unsound. The most that we can do is to collectively, as a community, learn to recognize patterns that frequently produce unsound claims.

Thus, this paper makes four primary contributions:

(1) It presents a principled framework that categorizes sins of reasoning as a mismatch between the scope of the claim and the scope of the evaluation. In doing so, it provides a high-level checklist for experimenters to use for avoiding unsound claims. It illustrates and clarifies the different sins with examples drawn from prior published work or the authors' own experiences.
(2) It encourages the community to think about sound claims and to publish their findings, particularly when it involves discovering new factors that lead to unsound claims. Each such finding increases the knowledge of the entire community and thus reduces the risk of producing unsound claims.
(3) It challenges the community by calling for a change to a culture that values idea papers just as much as evaluation papers. Specifically, this paper takes the position that producing an insightful evaluation is just as hard and just as commendable as coming up with a great new idea; both should be rewarded accordingly.
(4) It identifies common sins of exposition, which obscure the communication of claims and their underlying evaluations. Elimination of the sins of exposition is a prerequisite to being able to reason about the sins of reasoning.

3. GOALS OF THIS PAPER

This paper is an 'ideas paper' about empirical evaluation. As such, we purposefully do not provide an empirical evaluation nor do we survey papers that have committed sins of exposition or reasoning. Moreover, this paper assumes that all participants behave ethically; that is, we focus on unsound claims that come about even with the best intentions.

In empirical evaluations, unsound claims are costly because they misdirect a field: encouraging pursuit of fruitless endeavours and discouraging investigation of promising ideas. Yet, empirical computer science lacks a systematic approach with which to identify unsound claims. The goal of this paper is to address this need by providing a framework that allows us to identify unsound claims — and the errors that lead to them — more easily. Our hope is that this framework will provide reviewers with a systematic approach to avoid the promulgation of unsound claims about empirical evaluation and provide practitioners with a basis for self critique, curtailing unsound claims at their source.

4. SINS OF EXPOSITION

The sins of exposition hinder our goal of ensuring the soundness of claims because the sins obfuscate the claim and/or the evaluation. This section discusses the two sins of exposition and illustrates them with examples.

4.1. Sin of Inscrutability

The sin of inscrutability occurs when poor exposition obscures a claim. In other words, the readers' understanding of the claim may be different from the intent of the author. The reader cannot begin to assess the soundness of a claim if the claim itself is unclear. The sin of inscrutability is not an attempt to deceive (to do so would be unethical).

Inscrutability arises in three common forms: omission, ambiguity, and distortion. The most basic example of inscrutability is the omission of an explicit claim. In this case, the reader is left to either decide that there is no claim (the work is literally meaningless), or to divine an implied claim, reading between the author's lines. Thus the advice of standard guides on scientific writing holds true — writers should take care to be explicit about their claims. A second common example of inscrutability occurs when a claim is ambiguous. This leaves the reader unsure of the authors' intent and thus unable to discern the soundness of the claim. For example, the claim might use the phrase 'improved performance' but the context may leave it ambiguous as to whether this phrase refers to latency or throughput of the system. A third common example of inscrutability occurs when the claim is distorted, leaving the reader with a clear, but incorrect, impression of the author's intent. For example, poor exposition may result in a claim that suggests that (total) execution time is improved by 10%, but in fact the garbage collector is improved by 10% and total time is improved by only 1%. In each of these cases, inscrutability will make the claim hard or impossible to reconcile with the evaluation.

Because the claim is synthesized by the author, the sin of inscrutability, at least in theory, is avoidable: an author should always be able to accurately express their intent. As we shall see, this differs from the sin of irreproducibility, which concerns a natural, rather than synthetic, phenomenon and thus may be impossible to completely avoid.

4.2. Sin of Irreproducibility

The sin of irreproducibility occurs when poor exposition obscures the evaluation; that is, obscuring the steps, the data, or both. As with the sin of inscrutability, irreproducibility has three common manifestations: omission, ambiguity, or distortion. Omis-

sion has historically been a major cause of irreproducibility and has three sources: (i) space constraints often mean that not all steps are reported and data is only presented in digest form, (ii) an incomplete understanding of the factors that are relevant to the evaluation; and (iii) confidentiality. Ambiguity is a common source of irreproducibility: imprecise language and lack of detail can leave important elements of the evaluation ambiguous. Distortion occurs when poor exposition results in an unambiguous but incorrect account of the evaluation, such as the use of incorrect units.

While a diligent researcher can avoid distortion and ambiguity, omission is not entirely avoidable. The issue of space constraints has become less important in the age of digital libraries, online publishing, and cloud storage, where authors can provide comprehensive accounts of steps and data outside the confines of their primary publication. However, the other two issues related to omission are unavoidable in general.

First, the steps may be unbounded in the limit. While a research community can often tightly bound what it considers to be 'significant' factors in the steps, it cannot know with certainty every aspect of the empirical environment that may have significant bearing on the outcome of the evaluation. Experience shows that whole communities can ignore important, though non-obvious, aspects of the empirical environment, only to find out their significance much later, throwing into question years of published results [Ioannidis 2005]. This situation invites two responses: (i) authors should be held to the community's standard of what is known about the importance of the empirical environment, and (ii) the community should intentionally and actively improve this knowledge, for example, by promoting reproduction studies.

The second issue is that while some authors have the liberty to make full disclosure of their empirical environment, others are constrained by matters of propriety and confidentiality. Thus, there exists a tension between irreproducibility and the knowledge brought to the community by authors who work within such constraints. Perhaps this tension is best dealt with by full disclosure of the limitations of the exposition of the evaluation. Readers can then be clear about to the extent of which they must take the author's evaluation on faith and which matters are concretely communicated. Armed with this, the reader can make a clearer determination of the soundness of the authors' claims.

The rest of this section demonstrates the sin of irreproducibility with a concrete example. It demonstrates how a factor that the community considered irrelevant (and thus omitted from the exposition of the evaluation) turned out to be relevant, thus throwing into question years of published results.

Figure 3 illustrates the effect of changing the size of the Linux environment variables on the quantification of speedup due to gcc -O3 over gcc -O2 (i.e., to determine the benefit of gcc's most expensive optimizations). A point (x, y) on this graph says that the ratio of execution time with gcc -O2 to execution time with gcc -O3 is y if the environment size is x. Each point in this graph gives the arithmetic mean and standard deviation from 30 runs; the run-to-run variation was miniscule and thus the error bars show up as a dash through the circles. To collect this data, the authors started with an empty environment and then extended the environment a few bytes at a time.

Figure 3 demonstrates that changing the environment size has a significant effect on the speedup due to different optimization levels. This occurs because the size of the environment affects memory layout (e.g., starting address of the call stack) and thus the performance of the program, because different memory layouts interact with the numerous underlying hardware buffers (e.g., caches) differently. Thus, if an evaluation only explores one environment size and that size is left out of the description of the evaluation, we commit the sin of irreproducibility.

While it was well known that memory layout affects performance, it was not obvious to most researchers that environment variables could affect memory layout enough to

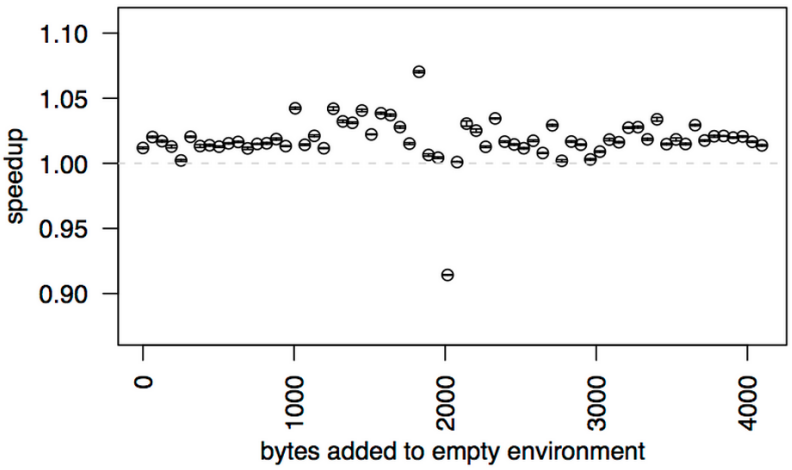

Fig. 3. Change in speedup as we vary the size of environment variables, which are ostensibly unrelated. Graph taken from [Mytkowicz et al. 2009].

affect performance. Therefore it was reasonable for experimenters to ignore the setting of environment variables in the evaluation when making a claim. While our example illustrates how memory layout subtly affects evaluation, the problem is more general: we commit the sin of irreproducibility whenever we exclude, in the description of the evaluation, any relevant control variables.

5. SINS OF REASONING

This section presents the sins of reasoning, which cause a claim to be unsound, and, for each sin, real examples found in the literature.

5.1. Sin of Ignorance

We commit the sin of ignorance when we make a claim that ignores elements of the evaluation that support a contradictory alternative claim. The sin of ignorance can manifest in a variety of ways including overlooking important data and overlooking a confounding variable that would provide an alternative explanation for observed behavior.

Often, the sin of ignorance is easy to see, such as when a claim ignores data outliers that undermine the claim. Other times the error is less obvious. The remainder of this section describes two non-obvious examples of the sin of ignorance

5.1.1. Ignoring data points. Deriving a claim from a subset of the data points may yield an unsound claim.

Figure 4 illustrates two different ways to summarize the data for different garbage collection algorithms. The graph on the left illustrates the best out of 30 runs for the benchmark *db* using one heap size. The claim derived from this graph states that the SemiSpace collector has significantly improved performance over all the other collectors, and that there is no substantial difference between CopyMS and GenCopy. The

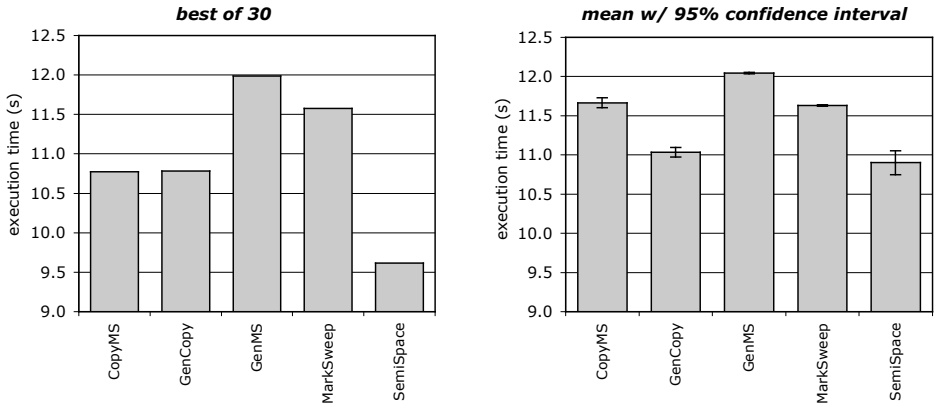

Fig. 4. Deriving a claim from a subset of the data commits the sin of ignorance. Graph from [Georges et al. 2007].

graph on the right illustrates the mean performance and the 95% confidence interval, which represents all the data points. The claim derived from this graph is significantly different, stating that although the SemiSpace collector has the best performance, its benefit over GenCopy is minimal, and CopyMS is significantly worse than GenCopy. The graph on the left commits the sin of ignorance when it derives a claim about performance that is based only on the best of 30 runs, because the claim singles out a particular data point without summarizing the rest. In general, we commit the sin of ignorance when we make claims about performance, but only use a subset of the data to derive the claim.

5.1.2. Ignoring data distribution. We often use statistics to summarize data and consequently derive a claim from that summarization. There are many statistics that one can calculate from a given set of data; however, commonly used statistical techniques make assumptions about the data (e.g., that the data is normally distributed). If a statistical technique's assumptions are not valid for our data, then applying this technique and deriving a claim from the statistics will yield an unsound claim, because the properties of our data are ignored.

Figure 5 illustrates a latency histogram for a component of Gmail [1]. The μ marks the arithmetic mean of the latency and $\mu - \sigma$ and $\mu + \sigma$ mark the points one standard deviation away from the mean. The x-axis of the graph is a log-scale and thus $\mu - \sigma$ and $\mu + \sigma$ are not equidistant from μ. From this data we make the claim that 68% of the requests will have a latency between the $\mu - \sigma$ and $\mu + \sigma$ (this is a property of normal data). This claim incorrectly assumes that the data is normal and ignores a key property of the collected data: the data distribution is bimodal (the histogram has two peaks corresponding to the slow and fast paths respectively). Consequently we see that a lot more than 68% of the data is one standard deviation away from the mean (for convenience we have marked the 68th percentile on the graph). Using the arithmetic mean and variance to derive the claim about Gmail performance commits the sin of ignorance because the statistics ignores key properties of the collected data. In general, we commit the sin of ignorance when we ignore the distribution of the underlying data

[1] One of the authors is a member of the Gmail team and was able to collect this data from live production traffic.

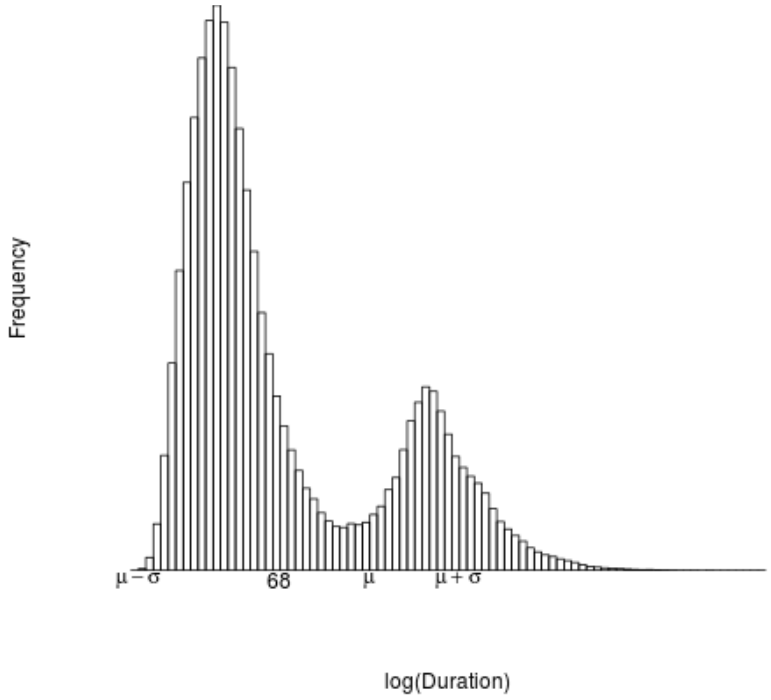

Fig. 5. A latency histogram for a component of Gmail. The x-axis is log-scale. The μ and σ are the arithmetic mean and standard deviation respectively. Using the arithmetic mean and standard deviation to derive a claim about Gmail performance is misleading because the mean does not capture the bi-modal or long tail nature of the data.

and yet use statistical methods (without checking the underlying assumptions about the data distribution) to make our claims.

Our example, as described above, is a cut-and-dried instance of the sin of ignorance. However, the committed sin is not always so obvious. For example, let us change the above example so that we calculate the mean and standard deviation on-the-fly (and thus never collect the raw data). In that case we commit two sins: (i) a sin of irreproducibility because our evaluation does not record a factor (the actual raw data and its distribution) that is essential to the claims we wish to make; and (ii) a sin of inappropriateness because we assume a normal distribution when making our claim even though there is no evidence for or against it in the evaluation.

5.1.3. Summary. In our experience, while the sin of ignorance seems obvious and easy to avoid, in reality it is far from that. Many factors in the evaluation that seem irrelevant to a claim may actually be critical to the soundness of the claim. As a community we need to work towards identifying these factors.

5.2. Sin of Inappropriateness

We commit the sin of inappropriateness when we derive a claim that is predicated on the presence of some fact; however, that fact is absent from the evaluation. For example, our claim may include: (i) performance, but the data only contains cache misses, (ii) a benchmark suite, but the data only covers a subset of the benchmarks, or (iii) scalability, but the benchmarks are not sufficiently parallel. If you have ever derived

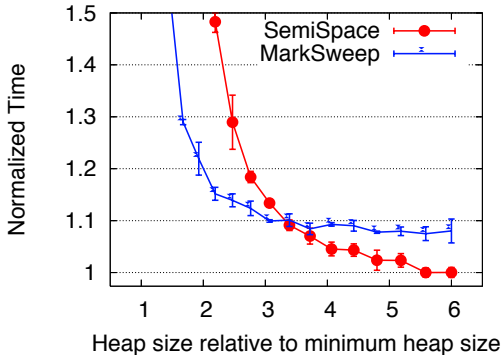

Fig. 6. Fixing the heap size can yield an incorrect result when evaluating GC. Graph taken from [Blackburn et al. 2008].

a claim from an evaluation and found that it did not apply to a real environment, you may have suffered from the sin of inappropriateness.

The sin of inappropriateness sounds obvious. However, in general, the sin of inappropriateness is not obvious. This section gives three examples to illustrate this.

5.2.1. Using inappropriate metrics. Energy and peak power are increasingly important concerns in many areas of computer science. Both power and energy have been particularly hard to accurately measure at the chip level. Unfortunately it is not uncommon for authors to measure a change in execution time but to claim a change in energy consumption. While execution time is often correlated with energy consumption, other factors (e.g., the nature of the computation) can also affect energy consumption. Thus, execution time does not support a claim about energy consumption.

Knowing that execution time is not always an accurate proxy for energy consumption is not obvious. Before this fact was pointed out by Martin and Siewiorek [2001], many papers on energy consumption suffered from this sin. We may commit the sin of inappropriateness when we use a proxy metric to make claims about another metric and the proxy metric does not always correlate to the other metric.

5.2.2. Misuse of independent variables. This section illustrates an example of deriving a claim from only one value of a variable as if that value represents all possible values. For many years, garbage collection results were typically reported for only one heap size, implying that the results applied to all heap sizes.

Figure 6 illustrates how the size of the heap affects the relative performance of different garbage collection implementations. A garbage collector that is best for one heap size may perform poorly for other sizes; thus, we commit the sin of inappropriateness if we derive a claim that is not limited by heap size, but the evaluation contains data for only one heap size.

Knowing that heap size affects performance for different garbage collection implementations is not necessarily obvious. Once it became known and published in various venues (including a book [Jones and Lins 1996]), it has now become normal to vary heap sizes in evaluating garbage collectors. Thus, exposing this common source of the sin of inappropriateness produced a positive effect on the community and effectively removed this particular sin from subsequent work.

5.2.3. Generalizing from biased sampling. We often use tools to perform measurements. However, these tools may themselves be subtly biased; if our claim generalizes from

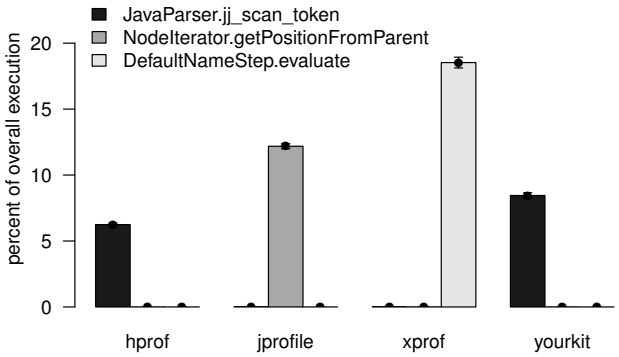

Fig. 7. Hotness of three methods in the pmd benchmark according to four profilers. The choice of profiler radically affects the outcome [Mytkowicz et al. 2010].

such biased data without accounting for the bias, the claim may commit the sin of inappropriateness.

For example, Figure 7 illustrates the hot method according to four different profilers for the pmd benchmark. The graph has four sets of bars, one for each profiler. Each set gives the CPU cycles spent in three important methods in the benchmark. We see that the different profilers significantly disagree: e.g., xprof considers evaluate to be a hot method consuming more than 18% of CPU cycles; however, the other three profilers do not identify this method as consuming any CPU cycles.

This disagreement comes about because the profilers use biased, rather than random, sampling (specifically the profilers used GC safe points for sampling, which are not randomly distributed [Mytkowicz et al. 2010; Buytaert et al. 2007]). We commit the sin of inappropriateness when we derive a claim from biased data, which consists of just a subset of points, but our claim does not acknowledge biased sampling. Knowing that these profilers use biased sampling is not obvious. Prior to these measurements, it was not widely known that Java profilers were biased and that the bias was significantly affecting claims derived from their output. We commit the sin of inappropriateness when we derive a claim about performance from biased data.

5.2.4. Summary. In our experience, while the sin of inappropriateness seems obvious and easy to avoid, in reality, it is far from that. Many factors that may be unaccounted for in evaluation may actually be important to derive a sound claim. As a community we need to work towards identifying these factors.

5.3. Sin of Inconsistency

While the sins of ignorance and inappropriateness are sins about what to include and what not to include in a claim, the sin of inconsistency is a sin of a faulty comparison. Specifically, we commit the sin of inconsistency when a claim compares two systems, but each system is evaluated in a way that is inconsistent with the other. For example, a claim states that an optimization is beneficial by comparing the performance of the optimized code on a server with the performance of the non-optimized code on a laptop. This comparison is inconsistent because the difference in performance may be due to the different measurement contexts (server versus laptop) and not due to the optimization. In addition to different measurement environments, we commit the sin of inconsistency when we derive a claim that is contingent on a comparison that uses (i) different metrics (e.g., wall clock versus cycles); (ii) different workloads; or (iii) differ-

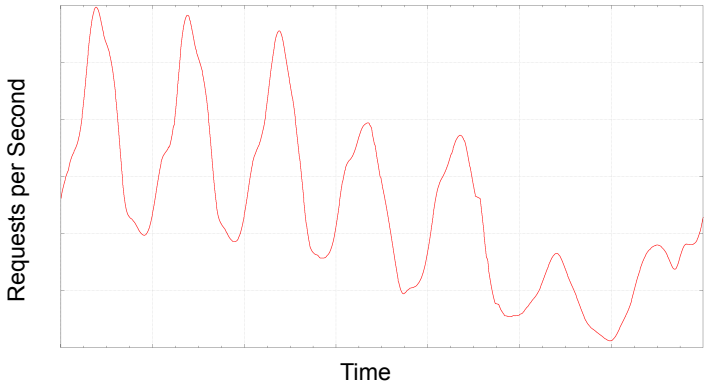

Fig. 8. Requests per second to Gmail over the course of a week. Evaluating an optimization by disabling it during the first half of the week and enabling it during the second half of the week commits the sin of inconsistency.

ent data analysis, (e.g., average run versus best run). If you have ever derived a claim from an improper comparison of data, you have suffered from a sin of inconsistency.

The sin of inconsistency sounds obvious; indeed most readers will readily find flaws in our example above. However, in general the sin of inconsistency is not obvious: this section will give examples to illustrate this.

5.3.1. Inconsistent hardware performance counters. When gathering data, we may use hardware performance counters, which count hardware events, to understand the behavior and performance of our system. However, the performance counters that are provided by different vendors may vary, and even the performance counters provided by the same vendor may vary across different generations of the same architecture. Comparing data from what may seem like similar performance counters within an architecture, across architectures or between generations of the same architecture, may result in the sin of inconsistency, because the hardware performance counters are counting different hardware events.

An example of an inconsistent comparison of hardware performance counters is to evaluate performance by comparing issued instructions with retired instructions on an out-of-order speculative machine. On an out-of-order speculative machine, there may be many more instructions issued than retired, because a speculative instruction may never retire. Therefore, deriving a claim by comparing issued to retired instructions to evaluate performance commits the sin of inconsistency, because the comparison may be evaluating how well the machine's speculation works, rather than determining a difference in performance.

5.3.2. Inconsistent workloads. The workload on a server application may vary from day to day and even from hour to hour. Therefore any evaluation of optimizing the performance of a server application must make sure that the same workload is used in the comparison with and without the optimization. If the performance of the server application during the weekend without optimizations is compared with the performance of the server application during the week with optimization to derive a claim, the claim may suffer from the sin of inconsistency.

Figure 8 illustrates workload variations in the context of Gmail. One of the authors is a member of the Gmail team and this graph from live production traffic shows the rate of incoming requests for Gmail over the course of the week (Monday-Sunday).

We see that the load varies significantly from day to day and even within the course of a day. If we evaluate an optimization by disabling it for the first half of the week and enabling it for the second half of the week we will end up with an inconsistent comparison. Consequently, any claim may reflect the difference in load over time and not the effect of the optimization.

5.3.3. Inconsistency in high performance computing. Bailey [2009] has identified sins of inconsistency in the high-performance computing community. We discuss a subset of them here.

Comparing the performance of two different high-performance algorithms, but using different floating-point precision for the arithmetic operations in the evaluation of each algorithm, commits the sin of inconsistency. This is because, on many systems, 32-bit computation rates are twice that of 64-bit rates. Therefore, the comparison is more about the effect of floating-point precision on execution time than about the algorithms.

Claims that compare the time to run a parallel algorithm on a dedicated system with the time to run a conventional (sequential) algorithm on a busy system commit the sin of inconsistency because the comparison may be more about the state of the underlying measurement contexts, a dedicated versus a busy system, than about the parallel versus conventional algorithms.

5.3.4. Summary. In our experience, while the sin of inconsistency seems obvious and easy to avoid, in reality it is far from that. Many artifacts that seem comparable may actually be inconsistent. As a community we need to work towards identifying these artifacts.

6. HOW TO USE OUR FRAMEWORK

There exists no framework or checklist that can completely eliminate unsound claims; after all, we see that other sciences, despite a much longer history than computer science, also suffer from unsound claims (e.g., [Ioannidis 2005]). Figure 9 expands upon Figure 1 to make this clear: our knowledge of the evaluation's data and steps is always imperfect: even experts cannot completely enumerate all of the factors that may affect or bias an evaluation; specifically, the 'Evaluation' box is dashed to indicate that even the experimenter may not have full knowledge of this. While our framework cannot eliminate unsound claims, it does bring up questions that authors and consumers (e.g., reviewers or artifact evaluation committees) should think about: we have found that thinking about these questions helps identify weaknesses in our claims. In this way, our framework reduces a 3rd order of ignorance [Armour 2000] (*I don't know something and I do not have a process to find out that I don't know it*) to a 2nd order of ignorance (*I don't know something but I do have a process to find out that I don't know it*).

For example, the author of an evaluation may ask the questions: (i) Have I considered all the data in my evaluation and not just the data that supports my claim (sin of ignorance)? (ii) Have I made any assumptions in forming my claim that are not justified by my evaluation (sin of inappropriateness)? (iii) When comparing my experimental evaluation to prior work, am I doing an apples-to-apples comparison (sin of inconsistency)? (iv) Have I adequately described everything I know to be essential for getting my results (sin of irreproducibility)? (v) Have I unambiguously and clearly stated my claim (sin of inscrutability)?

7. A CALL FOR CULTURE CHANGE

The five sins are not unique to computer science; other sciences also suffer from them also. Indeed other sciences frequently publish papers that refute prior work (e.g. [Ioannidis 2005]). Thus, while it is difficult to eliminate these sins, our community can

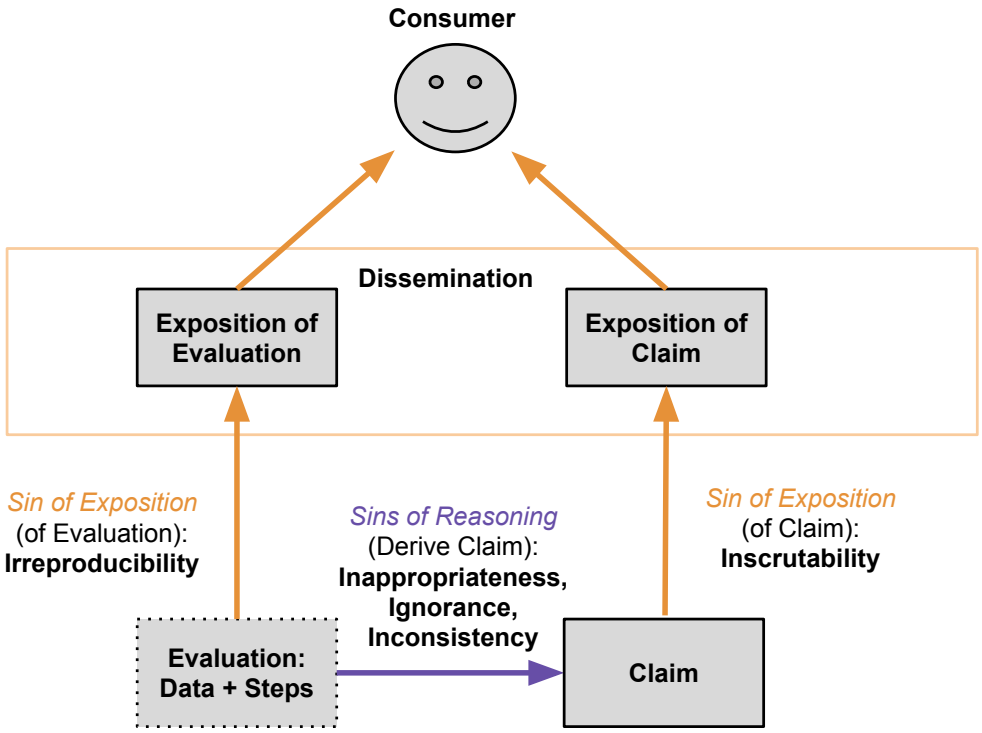

Fig. 9. Enhancement of Figure 1, which illustrates the sins of exposition and sins of reasoning. The sins of exposition obfuscate the evaluation and/or the claim, preventing an observer from determining the soundness of the claim. The sins of reasoning interfere with the derivation of the claim, resulting in an unsound claim.

strive towards a culture that recognizes exemplary evaluation papers. By 'exemplary' we mean papers that go out of their way to conduct careful and insightful evaluations.

More concretely, we can place papers in our field in a two dimensional space (Figure 10). The first dimension is 'novelty'; papers score highly in this dimension if they present novel ideas. The second dimension is 'quality of evaluation'; papers score highly in this dimension if they present careful and insightful evaluation.

Papers in the (low-novelty, low-evaluation) cell do not present new algorithms nor present insightful empirical evaluation; thus these papers are justifiably rejected from our publication venues. Papers in the (high-novelty, high-evaluation) cell present exciting new algorithms or ideas and a compelling empirical evaluation. Such papers are rare and their scope is often much larger than a single paper. Papers in (middle-novelty, middle-evaluation) cell are a 'safe bet': they have modestly novel algorithms or ideas and a reasonable (but not fully convincing) evaluation. Thus these papers are not an obvious failure along either dimension.

While 'safe bet' papers are often easy to publish, these papers tend to be incremental; thus rarely open up new opportunities and fields.

In our experience, our community rarely publishes papers that are in the low-novelty, high-evaluation and high-novelty, low-evaluation cells. We believe that the former get rejected because they do not present any new algorithms. We believe that the latter get rejected because they present algorithms or ideas without empirical evidence for their efficacy.

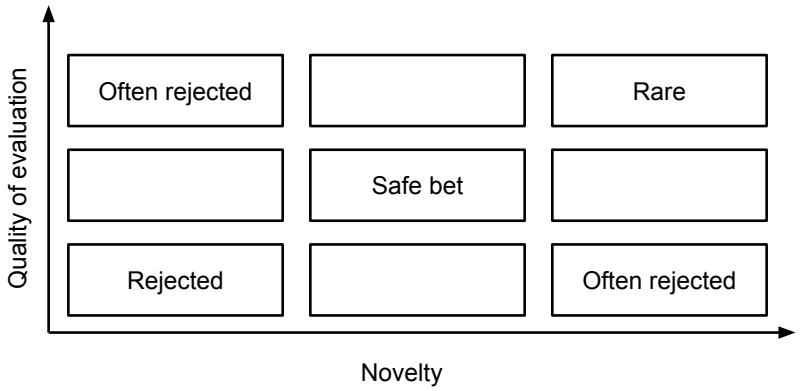

Fig. 10. Types of a papers. We can categorize papers along two dimensions: (1) 'novelty', where papers score highly in this dimension if they present novel ideas; (2) 'quality of evaluation', where papers score highly in this dimension if they present careful and insightful experimental evaluations.

We believe that these two cells are important and that our community should be encouraging papers in these categories.

Papers that have insightful empirical evaluations but no new algorithms are worth publishing. Insightful evaluations often (i) expose new opportunities for exploration; (ii) provide confidence in the validity or invalidity of ideas; and (iii) identify sins that may be so far unknown by the community; e.g., they may identify new factors that affect the outcome of evaluations and thus contribute to sins of irreproducibility or ignorance. Thus, these papers are key to the advancement of our field. We contend that a paper that has an exemplary insightful evaluation and no new algorithms is superior to a paper that has an exemplary evaluation and gratuitously includes a low-novelty algorithm: the low-novelty algorithm may have been placed in the paper to satisfy conference reviewers and it may actually mislead readers away from the true impact of the empirical work.

Papers that have highly novel algorithms and ideas but no real empirical evaluation are also worth publishing as long as (i) they have theoretical or other arguments for the likely usefulness of the algorithms, and (ii) do not make unsound claims.

These papers expose new ways of thinking about problems and open up entire fields of exploration. We contend that a paper that has highly novel algorithms but no empirical evaluation is superior to a paper with highly novel algorithms and unsound claims: the unsound claims may mislead us as to the true value of the ideas. We recognize that quick and dirty evaluations have a place in a researcher's toolbox: they may be helpful in determining which of many different avenues to explore. However, an unsound claim has no place in published research.

8. RELATED WORK

Our paper is related to several efforts to improve empirical evaluation and improve the science in computer science.

8.1. Artifact Evaluation Committees

Several programming languages and software engineering conferences recently introduced so-called 'Artifact Evaluation Committees' (AECs) [Krishnamurthi 2013]. These

committees complement the program committees, with the goal 'to empower others to build on top of the contributions of a paper' [Hauswirth and Blackburn 2013]. Their main purpose is to evaluate the artifacts underlying the submitted papers along four dimensions: their consistency with the claims formulated in the paper, their completeness, their documentation, and the ease with which they can be reused. While the costs and benefits of AECs have been debated [Krishnamurthi et al. 2013], most major programming languages conferences now include AECs. The 'consistency' as evaluated by the AEC is related to our framework. In fact, it is affected by all the sins we describe (not just by the sin of inconsistency). While the submitted artifact supplements what is in the paper, in general, it cannot fully reflect the evaluation as conducted by the original author. Our framework can be used by an AEC in their evaluation of an artifact and a paper to identify sins.

Other areas in computer science introduced similar committees. For example, the SIGMOD conference uses so-called 'Repeatability' committees [Bonnet et al. 2011; Manegold et al. 2010], to verify the evaluations published in the conference. They have two goals: evaluate the 'repeatability' (independently reproducing the evaluations) and the 'workability' (exploring changes to evaluation's parameters).

8.2. Separating Ideas from Evaluations

Our call for a culture change in Section 7 reiterates that the idea and the evaluation are two separate dimensions of a paper. We encourage papers with novel ideas but no evaluation and papers with strong evaluations without novel algorithms. Both should be viewed as contributions and accepted in our conferences.

Dittrich [2011] goes even further. He partitions traditional papers into 'paper bricks', and considers each such brick an independent contribution. Proposed bricks include 'introduction', 'problem statement', 'high level solution idea', 'details', and 'performance evaluation'. Thus, it clearly separates the evaluation from the idea, partitions the idea further, and introduces the identification of the problem (which might be driven by an experiment or observational study) as a separate contribution.

8.3. Literate Experimentation

Singer [2011] proposes an approach to empirical evaluation that tightly couples the experiment and the paper's description of the experiment: both are automatically derived from the same specification. Essentially, the specification of the experiment and its explanation are interleaved as part of the source text of the paper. A toolchain reads the specification from that source, and either generates the paper, or runs the experiment. This reduces inconsistencies between the description of the experiment in the paper, and the actual experiment. It also makes the experiment more repeatable. However, not all aspects of the experiment can be captured in this way (e.g. the temperature in the room while running the experiment), and thus the approach still is susceptible to the sins in our framework.

While Singer introduces the idea of literate experimentation and provides an implementation based on scripts and targeting virtual machine research, Schulte et al. [2012] provide an implementation of a very similar approach based on emacs org-mode.

8.4. Independent Reproduction

Baker [2012] and Begley [2012] report on the commercialization of scientific reproduction studies. The company Science Exchange offers the 'Reproducibility Initiative', a way for scientists publishing a paper to have their study reproduced. Scientists of the original study pay for having the study reproduced by a second, independent lab selected by Science Exchange. The two research teams then publish a second paper together with the results of the reproduction study. Begley's article quotes Glenn Begley,

the former head of global cancer research at Amgen, as saying 'There are too many incentives to publish flashy, but not necessarily correct, results'.

Both de Oliveira et al. [2013] and Keahey and Desprez [2012] present computational platforms that are available to a community of users to collect empirical results. These platforms can be used for reproducibility studies.

Although the evaluation setup, in which results are collected, is an essential aspect of an evaluation, our framework focuses on the evaluation after the results have already been collected.

8.5. Reproducibility

Reproducing empirical results is the foundation of science. If a result can't be reproduced, then it can't be used for prediction. There has been significant focus on the failure of reproducibility of empirical results both in computer science (Stodden et al. [2013] and Bailey et al. [2014]) and in other fields (Lehrer [2010] and eco [2013]), and how to fix it (Touati et al. [2013]). Our framework identifies, at a high level of abstraction, the inability to reproduce empirical results as the sin of irreproducibility. In addition, our framework identifies four other ways an empirical evaluation may be unsound.

In Section 1.2, we point out that reproducibility is hard to ensure, because an unbounded number of factors in the environment may affect the outcome of the evaluation in subtle ways. This point is important to emphasis. Nevertheless, reproducibility is an important goal we must continue to strive for. There have been community efforts to improve reproducibility. For example, Fursin [2015] promotes a venue for open publication.

8.6. Examples of Sins of Reasoning

Norvig [2012] and Vitek and Kalibera [2011] present multiple examples of the sins of reasoning in experimental design and interpretation of results. Our framework captures each of these examples at a higher level of abstraction. In Section 1.1 we prove that our sins of reasoning cover all possibilities and thus are principled and complete. In addition, Vitek and Kalibera [2011] propose a number of recommendations to improve the way we do empirical evaluation.

8.7. Experimental Design

Pieterse and Flater [2014] discuss the pitfalls of experimental design, which is needed to evaluate software performance, and provide a software performance experiment life cycle process.

8.8. Challenging Cultural Norms

Nowatzki et al. [2015] challenge the field of computer architecture to reconsider how architects evaluate their work. They question their field's preoccupation with cycle-accurate simulation, and propose that the evaluation technique reflect the 'footprint' of the subject of the evaluation — in other words, they propose that the scope of the evaluation match the scope of the claim. They offer guidance on how to avoid pitfalls of simulation-based evaluation of architectural innovations.

9. CONCLUSIONS

An evaluation, in isolation, is never wrong. It may be uninteresting or thoroughly limited in scope, but it is still not wrong. However, when we derive claims from an evaluation, we may commit errors that render the claim unsound. This paper shows that many common errors fall into two broad classes of error: sins of exposition and sins of reasoning.

To help experimenters identify unsound claims and derive sound claims, this paper makes two contributions. First, it provides a framework to understand and categorize common mistakes that lead to unsound claims. The framework provides practitioners with a principled way of critiquing the integrity of their own work and the work of others. We hope that this will help individuals conduct better science and encourage a cultural shift in our research community to identify and promulgate sound claims. Second, it gives examples for each category of sin; these examples draw from real sins that the authors have observed in their own work and in the literature. These examples provide evidence that these sins occur, and that these sins are therefore not apparently well understood.

This paper does not present a silver bullet: empirical evaluation is difficult and even the most seasoned evaluator may inadvertently commit a sin. Our hope is that this framework will provide reviewers with a systematic approach to avoid the promulgation of unsound claims about empirical evaluation and provide practitioners with a basis for self critique, curtailing unsound claims at the source.

REFERENCES

2013. Unreliable research. Trouble at the lab. *The Economist* (19 October 2013). http://www.economist.com/news/briefing/21588057-scientists-think-science-self-correcting-alarming-degree-it-not-trouble

Phillip G. Armour. 2000. The Five Orders of Ignorance. *Commun. ACM* 43, 10 (Oct. 2000), 17–20. DOI:http://dx.doi.org/10.1145/352183.352194

David H. Bailey. 2009. Misleading performance claims in parallel computation. In *46th Annual Design Automation Conference*. ACM, ACM Press, New York, NY, 528–33. http://dx.doi.org/10.1145/1629911.1630049

David H. Bailey, Jonathan M. Borwein, and Victoria Stodden. 2014. Facilitating reproducibility in scientific computing: Principles and practice. (30 June 2014). http://www.davidhbailey.com/dhbpapers/reprod.pdf

M. Baker. 2012. Independent labs to verify high-profile papers: Reproducibility Initiative aims to speed up preclinical research. *Nature : News* (14 August 2012). doi:10.1038/nature.2012.11176

Sharon Begley. 2012. More trial, less error - An effort to improve scientific studies. *Reuters* (14 August 2012). http://www.reuters.com/article/2012/08/14/us-science-replication-service-idUSBRE87D0I820120814

Stephen M. Blackburn, Kathryn S. McKinley, Robin Garner, Chris Hoffmann, Asjad M. Khan, Rotem Bentzur, Amer Diwan, Daniel Feinberg, Daniel Frampton, Samuel Z. Guyer, Martin Hirzel, Antony Hosking, Maria Jump, Han Lee, J. Eliot B. Moss, Aashish Phansalkar, Darko Stefanovik, Thomas VanDrunen, Daniel von Dincklage, and Ben Wiedermann. 2008. Wake Up and Smell the Coffee: Evaluation Methodology for the 21st Century. *Commun. ACM* 51, 8 (Aug. 2008), 83–89. DOI:http://dx.doi.org/10.1145/1378704.1378723

Philippe Bonnet, Stefan Manegold, Matias Bjørling, Wei Cao, Javier Gonzalez, Joel Granados, Nancy Hall, Stratos Idreos, Milena Ivanova, Ryan Johnson, David Koop, Tim Kraska, René Müller, Dan Olteanu, Paolo Papotti, Christine Reilly, Dimitris Tsirogiannis, Cong Yu, Juliana Freire, and Dennis Shasha. 2011. Repeatability and Workability Evaluation of SIGMOD 2011. *SIGMOD Rec.* 40, 2 (Sept. 2011), 45–48. DOI:http://dx.doi.org/10.1145/2034863.2034873

Frederick P. Brooks, Jr. 1996. The Computer Scientist As Toolsmith II. *Commun. ACM* 39, 3 (March 1996), 61–68. DOI:http://dx.doi.org/10.1145/227234.227243

D. Buytaert, A. Georges, M. Hind, M. Arnold, L. Eeckhout, and K. De Bosschere. 2007. Using hpm-sampling to drive dynamic compilation. In *Proceedings of the 22nd annual ACM SIGPLAN conference on Object-oriented programming systems and applications*. ACM, ACM Press, New York, NY, 553–568. http://dx.doi.org/10.1145/1297105.1297068

Augusto Born de Oliveira, Jean-Christophe Petkovich, Thomas Reidemeister, and Sebastian Fischmeister. 2013. DataMill: Rigorous Performance Evaluation Made Easy. In *Proceedings of the 4th ACM/SPEC International Conference on Performance Engineering (ICPE '13)*. ACM, New York, NY, USA, 137–148. DOI:http://dx.doi.org/10.1145/2479871.2479892

Jens Dittrich. 2011. Paper Bricks: An Alternative to Complete-story Peer Reviewing. *SIGMOD Rec.* 39, 4 (May 2011), 31–36. DOI:http://dx.doi.org/10.1145/1978915.1978923

Grigori Fursin. 2015. Enabling collaborative, systematic and reproducible research and experimentation in computer engineering with an open publication model. website. (2015). http://ctuning.org/cm/wiki/index.php?title=Reproducibility

A. Georges, D. Buytaert, and L. Eeckhout. 2007. Statistically rigorous Java performance evaluation. In *Proceedings of the 22nd ACM SIGPLAN conference on Object-Oriented Programming, Systems, Languages and Applications*. ACM Press, New York, NY, 57–76. http://dx.doi.org/10.1145/1297105.1297033

Matthias Hauswirth and Stephen M. Blackburn. 2013. Artifact Evaluation Artifact. web site. (2013). http://evaluate.inf.usi.ch/artifacts/aea

J.P.A. Ioannidis. 2005. Contradicted and initially stronger effects in highly cited clinical research. *American Medical Association* (2005), 218–228.

Richard Jones and Rafael Lins. 1996. *Garbage Collection: Algorithms for Automatic Dynamic Memory Management*. Wiley. http://dl.acm.org/citation.cfm?id=236254

Kate Keahey and Frdric Desprez. 2012. *Supporting Experimental Computer Science*. Technical Report MCS Technical Memo 326. Argonne National Laboratory (ANL). http://www.nimbusproject.org/downloads/Supporting_Experimental_Computer_Science_final_draft.pdf

Shriram Krishnamurthi. 2013. Artifact Evaluation for Software Conferences. *SIGPLAN Not.* 48, 4S (July 2013), 17–21. DOI:http://dx.doi.org/10.1145/2502508.2502518

Shriram Krishnamurthi, James Noble, and Jan Vitek. 2013. Should Software Conferences Respect Software?. In *Proceedings of the 2013 Companion Publication for Conference on Systems, Programming, & Applications: Software for Humanity (SPLASH '13)*. ACM, New York, NY, USA, 71–72. DOI:http://dx.doi.org/10.1145/2508075.2516929

Jonah Lehrer. 2010. The Truth Wears Off: Is there something wrong with the scientific method? *The New Yorker* (13 December 2010). http://www.newyorker.com/magazine/2010/12/13/the-truth-wears-off

S. Manegold, I. Manolescu, L. Afanasiev, J. Feng, G. Gou, M. Hadjieleftheriou, S. Harizopoulos, P. Kalnis, K. Karanasos, D. Laurent, M. Lupu, N. Onose, C. Ré, V. Sans, P. Senellart, T. Wu, and D. Shasha. 2010. Repeatability & Workability Evaluation of SIGMOD 2009. *SIGMOD Rec.* 38, 3 (Dec. 2010), 40–43. DOI:http://dx.doi.org/10.1145/1815933.1815944

T. L. Martin and D. P. Siewiorek. 2001. Nonideal battery and main memory effects on CPU speed-setting for low power. *IEEE Transactions on Very Large Scale Integration Systems* 9 (Feb 2001), 29–34. http://dx.doi.org/10.1109/92.920816

Todd Mytkowicz, Amer Diwan, Matthias Hauswirth, and Peter F. Sweeney. 2009. Producing wrong data without doing anything obviously wrong!. In *Proceedings of the 14th international conference on Architectural support for programming languages and operating systems*. ACM, ACM Press, New York, NY, 265–276. http://dx.doi.org/10.1145/1508244.1508275

Todd Mytkowicz, Amer Diwan, Matthias Hauswirth, and Peter F. Sweeney. 2010. Evaluating the accuracy of Java profilers. In *Proceedings of the 2010 ACM SIGPLAN conference on Programming language design and implementation*. ACM, ACM Press, New York, NY, 187–197. http://dx.doi.org/10.1145/1806596.1806618

Peter Norvig. 2012. Warning Signs in Experimental Design and Interpretation. (2012). http://norvig.com/experiment-design.html

Tony Nowatzki, Jaikrishnan Menon, Chen-Han Ho, and Karthikeyan Sankaralingam. 2015. Architectural Simulators Considered Harmful. *IEEE Micro* (2015).

Vreda Pieterse and David Flater. 2014. *The ghost in the machine: dont let it haunt your software performance measurements*. Technical Report Technical Note 1830. NIST. http://dx.doi.org/10.6028/NIST.TN.1830

Eric Schulte, Dan Davison, Thomas Dye, and Carsten Dominik. 2012. A Multi-Language Computing Environment for Literate Programming and Reproducible Research. *Journal of Statistical Software* 46, 3 (25 1 2012), 1–24. http://www.jstatsoft.org/v46/i03

Jeremy Singer. 2011. A Literate Experimentation Manifesto. In *Proceedings of the 10th SIGPLAN Symposium on New Ideas, New Paradigms, and Reflections on Programming and Software (Onward! 2011)*. ACM, New York, NY, USA, 91–102. DOI:http://dx.doi.org/10.1145/2048237.2048249

Victoria Stodden, Jonathan Borwein, and David H. Bailey. 2013. 'Setting the default to reproducible' in computational science research. *SIAM News* 46, 5 (June 2013), 4–6. http://www.siam.org/news/news.php?id=2078

Sid-Ahmed-Ali Touati, Julien Worms, and Sbastien Briais. 2013. The Speedup-Test: A Statistical Methodology for Program Speedup Analysis and Computation. *Journal of Concurrency and Computation: Practice and Experience* 25, 10 (July 2013), 1410–1426. http://dx.doi.org/10.1002/cpe.2939http://hal.inria.fr/hal-00764454

Jan Vitek and Tomas Kalibera. 2011. Repeatability, Reproducibility, and Rigor in Systems Research. In *Proceedings of the Ninth ACM International Conference on Embedded Software (EMSOFT '11)*. ACM, New York, NY, USA, 33–38. DOI:http://dx.doi.org/10.1145/2038642.2038650