Recombinable Benchmarks and Tools

Simon Dierl[1] and Falk Howar[1,2]

[1] TU Dortmund University, Dortmund, Germany
{simon.dierl, falk.howar}@tu-dortmund.de
[2] Fraunhofer ISST, Dortmund, Germany

1 Introduction

In many disciplines, including Software Engineering, research involves the implementation of ideas and algorithms into (prototypical) *tools*. Claims about the correctness or performance of the algorithms are evaluated by executing these tools on *benchmarks*, i.e., collections of data and accompanying metadata.

The creation of *replication packages*, i.e., peer-reviewed, long-term archived bundles, usually containers, accompanying publications, ensures that these experiments can be re-run by other researchers. W.r.t. to the FAIR [5] and FAIR4RS [1] principles, accessibility is guaranteed by long-term archival, while findability depends on the package's metadata.

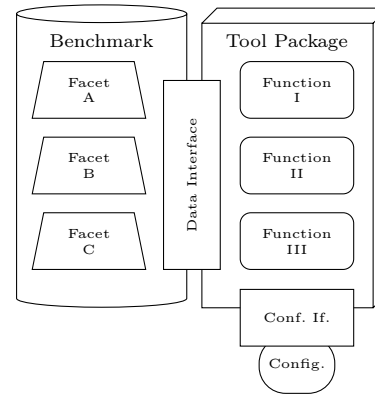

Figure 1. A recombinable learning tool and benchmark.

However, research building upon prior ideas requires not only repetition of past experiments, but *recombination* of "historic" tools with new or extended benchmarks as well as new tools with previously used benchmarks. The former enables the study of tools' "evolution" and the latter enables comparisons between competing tools based on large benchmark sets. This can not be achieved via replication packages, since they are closed systems.

We propose a novel approach [4] to packaging benchmarks and tools that enables recombination inspired by, i.a., ETI [3] and SV-COMP's tool format [2], sketched in Fig. 1: tools and benchmarks are packaged (and distributed) independently. Combination for a study uses standardized interfaces instead of creating a new artifact. For tools, this necessitates a container format that allows users to make benchmarks, configurations, etc. available to the tool. For benchmarks, we do not envision study-specific data, but long-lived data sets aggregating the greatest amount of data possible that can be used in a wide variety of fields. Recombination requires standardized basic exchange formats, i.e., the tool container format and descriptors for tools and benchmarks. We do, however, not prescribe the actual *data* formats. Recombinable tools and benchmarks will, by definition, be interoperable and reusable, while descriptors increase their findability w.r.t. to the FAIR and FAIR4RS guidelines.

2 Proposed Specification

Our proposed *abstract* specification defines an abstract approach to creating benchmarks, a standardized tool container format, and a descriptor language for both. As far as possible, all are based on established technologies.

Benchmarks. Ideally, a benchmark is stored using an structured, hierarchical data format such as JSON or YAML. Then, a concrete specification of the format must define the *syntax* (i.e., the data format) and the *semantics*. We propose to define the syntax via a media type. For the semantics, we must address the issue of storing and describing many different, but independent types of information present in the benchmark. We call these *facets*. We can then schematize each individual facet using JSON Schema as long as each facet's data is decoupled (e.g., by being stored in different fields of an associative array). The benchmark as such is then described by its media type and *all* facets' schemata it satisfies.

Tools. For tools, we recommend the use of the Singularity container format. As opposed to other container frameworks, e.g. Docker, this is designed to a) store containers in a single file and b) allows for easy file system integration, yielding a simple interface to inject benchmarks and configuration files. In addition, Singularity is already a popular technology in HPC.

Descriptors. Our proposed descriptor links benchmarks and tools by associating each with an accompanying `rereso-benchmark.yml` or `rereso-tool.yml`, respectively. Aside from commonly used metadata such as name, license, or references, benchmarks specify their formats as a combination of media type and schemata. Tools, however, describe their modes of invocation via input and output formats, each defined as a media type-schemata tuple. This also enables future work on automated search and type checking on, e.g., tool pipelines.

Acknowledgments. Funded by the German Research Foundation (DFG) – project numbers 442146713 (NFDI4Ing); 495857894 (STING).

References

1. Barker, M., et al.: Introducing the FAIR Principles for research software. Sci. Data **9**(1), 622. https://doi.org/10.1038/s41597-022-01710-x
2. Beyer, D., Strejček, J.: Improvements in software verification and witness validation: SV-COMP 2025. In: Proc. TACAS. pp. 151–186 (2025). https://doi.org/10.1007/978-3-031-90660-2_9
3. Braun, V., et al.: The ETI Online Service in action. In: Proc. TACAS. pp. 439–443 (1999). https://doi.org/10.1007/3-540-49059-0_31
4. Dierl, S., Howar, F.: Towards a specification for recombinable benchmarks and software tools. ing.grid preprint, https://preprints.inggrid.org/repository/view/55/
5. Wilkinson, M.D., et al.: The FAIR Guiding Principles for scientific data management and stewardship. Sci. Data **3**(1), 160018. https://doi.org/10.1038/sdata.2016.18