SDEval: A Benchmarking Toolkit for Computer Algebra

Hans-Gert Gräbe[1] Albert Heinle[2] Viktor Levandovskyy[3]

[1]Institut für Informatik, Leipzig University, Germany
[2]David R. Cheriton School of Computer Science, University of Waterloo, Canada
[3]Lehrstuhl D für Mathematik, RWTH Aachen University, Germany

Observations

We observed the following in the Computer Algebra community.

- There are no standards to reliably reconstruct and verify computations in research articles.
- For a given computation problem in the computer algebra community, there is mostly no standard test set defined to examine the quality of a new implementation.
- A variety of Computer Algebra Systems (CAS) is available with distinct syntaxes and strengths.

⇒ One CAS might fail to do a certain computation, while another one succeeds.

⇒ Even though attempts are made to have a unified interface (e.g. Sage), researchers still need to be accustomed to many CAS to get full access to the assistance available through CAS implementations.

Introducing SDEval

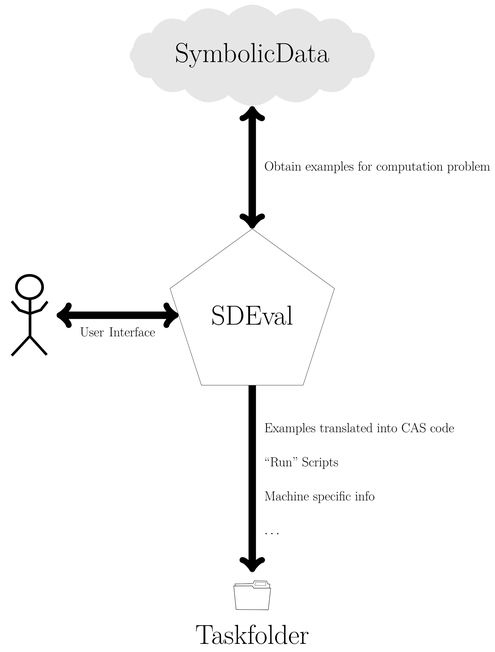
SDEval [3] is a benchmarking toolbox built on top of the SymbolicData database [1, 2]. Some of its main goals are

- providing an easy way of translating existing entries in SymbolicData into executable code of computer algebra systems,
- providing a feasible way of trustfully reproducing and verifying computation results from current research papers,
- meeting the particularities of benchmarking in the field of computer algebra and
- being flexible in order to be applicable across different communities.

SDEval targets, among others, two groups of researchers:

- Those who want to try out different CAS to find a solution to a problem that appeared in his/her research.
- Those who have created a new implementation for a certain computer algebra problem and wish to compare it to existing ones.

Create



Run

The Taskfolder contains a script called `runTasks.py`. It can be run using optional parameters.

Example

```
runTasks.py -c3600 -m1000000000 -j4
```

- `-c3600`: Stop any script after using more than 1h CPU time.
- `-m1000000000`: Stop any script using more than 1GB of the memory.
- `-j4`: run up to four scripts in parallel.

Furthermore, we have the following features:

- Visualization of the status as HTML files.
- User can manually terminate a CAS without having to restart the whole process.
- An interface for scripts interpreting the output of the CAS is given.
- The user can customize the computation by altering a preference-file given in XML format.
- Run-part independent from creation-part. One can write a taskfolder containing one's own code and use our scripts to run and monitor them.

The Taskfolder – An Easy Way to Reproduce Results.

- The taskfolder has the following structure:

```
+ TaskFolder
| - runTasks.py //For Running the task
| - taskInfo.xml //Saving the Task in XML Structure
| - machinesettings.xml//The Machine Settings in XML form
| + classes //All classes of the SDEval project
| + casSources //Folder containing all executable files
| | + SomeProblemInstance1
| | | + ComputerAlgebraSystem1
| | | | - executablefile.sdc //Executable code for CAS
| | | | - template_sol.py //Script to analyze the output of the CAS
| | | + ComputerAlgebraSystem2
| | | | - executablefile.sdc
| | | + ...
| | + SomeProblemInstance2
| | | + ...
| | + ...
```

- After executing `runTasks.py`, a subfolder `results` is added to the taskfolder.
- The taskfolder, containing the results, can then be published on a website.
- In order to reproduce the results, one solely has to adapt the commands to call the respective CAS inside `machinesettings.xml` to one's own machine and execute `runTasks.py` afterwards.
- The taskfolder structure and scripts are not bound to computer algebra systems. It can be used for any scripts and any software ⇒ SDEval can be used across communities.

Directions for the Future

- Support more computation problems and CAS.
- Communication with different communities about further use-cases and feature-needs for SDEval.
- Provide meaningful output-interpretation scripts in the distribution of SDEval. This is challenging since
  - outputs coming from algorithms in Computer Algebra are often not unique and
  - the evaluation of the correctness of an output is often not trivial and sometimes even subject of ongoing research.
- Preaching the practice of publishing taskfolders, so that computations in the literature can be verified.

References

[1] Gräbe, H.-G., Nareike, A., Johanning, S. (2014). The SymbolicData Project - from Data Store to Computer Algebra Social Network. Computeralgebra Rundbrief 55 (October 2014).

[2] Gräbe, H.-G., Nareike, A., and Johanning, S. (2014) The SymbolicData Project–Towards a Computer Algebra Social Network. *Workshop and Work in Progress Papers at CICM 2014 in CEUR-WS.org* vol. 1186 (2014)

[3] Heinle, A. and Levandovskyy, V. (2015). The SDEval Benchmarking Toolkit. *ACM Commun. Comput. Algebra*, 49(1):1–9. DOI http://doi.acm.org/10.1145/2768577.2768578

Acknowledgements

We thank the "Deutsche Forschungsgesellschaft" (DFG) for partial financial support for the development of SDEval in the context of the "DFG Priority Project SPP 1489: Algorithmic and Experimental Methods in Algebra, Geometry and Number Theory."

Resources

- http://www.symbolicdata.org
- https://github.com/ioah86/symbolicdata
- https://www.youtube.com/watch?v=CctmrfisZso