

Empirical Evaluation of $\{log\}$'s Decision Procedure for Binary Relations

Maximiliano Cristiá

Gianfranco Rossi

The VM for performing the artifact evaluation can be downloaded from <https://www.dropbox.com/s/8a0t3a88sv2c7xs/cav2016-113.ova?dl=0>
All the information is in directory `paper113`.

$\{log\}$ (read ‘setlog’) is a Constraint Logic Programming language implementing a decision procedure for finite, unbounded sets and binary relations. The decision procedure for sets has been extensively discussed [DPPR00]. The decision procedure for binary relations is the subject of the paper “A Decision Procedure for Sets, Binary Relations and Partial Functions” (#113).

The decision procedure takes the form of a rewriting system. This rewriting system has been proved to be sound, complete and terminating (Section 3). We can say that the rewriting system *works in theory*. Sect. 5 presents an empirical evaluation showing that the decision procedure also *works in practice*. The rewrite rules dealing with partial functions have already been evaluated [CRF15]. Therefore, this evaluation is focused on the rules for binary relations.

Hence, we put under revision the empirical evaluation performed in Sect. 5. Our evaluation consists in executing $\{log\}$ on 300 goals involving at least one binary relation and at least one of the 9 supported relational operators. These goals are the result of modifying some base goals in different ways as explained in Sect. 5. The base goals have been taken from the standard partitions proposed by the Test Template Framework [SC96, CAF⁺14] for the relational operators of the Z mathematical toolkit [Spi92].

Each goal is a formula of the language for sets and binary relations implemented by $\{log\}$. For practical reasons, each goal is saved in a separated file (300 files) containing, besides, a few more lines of Prolog code to load $\{log\}$ and take the execution time. The extension of these files is `p1`.

We also provide a `bash` script (`runAE`) that automatically runs all the goals, saves each result in a separated file (with extension `out`), summarizes all the results in a \LaTeX table and compiles this \LaTeX file into a PDF file. Due to space restrictions the table included in the paper is a compact version of the table generated by the script. See a more detailed version in `experiments.pdf`.

We claim that $\{log\}$ gives meaningful answers (i.e. `sat` or `unsat`) for 283 goals, that is the 94%, with a 10s timeout (per goal) and on a standard laptop.

References

- [CAF⁺14] Maximiliano Cristiá, Pablo Albertengo, Claudia S. Frydman, Brian Plüss, and Pablo Rodríguez Monetti. Tool support for the Test Template Framework. *Softw. Test., Verif. Reliab.*, 24(1):3–37, 2014.
- [CRF15] Maximiliano Cristiá, Gianfranco Rossi, and Claudia S. Frydman. Adding partial functions to constraint logic programming with sets. *TPLP*, 15(4-5):651–665, 2015.
- [DPPR00] Agostino Dovier, Carla Piazza, Enrico Pontelli, and Gianfranco Rossi. Sets and constraint logic programming. *ACM Trans. Program. Lang. Syst.*, 22(5):861–931, 2000.
- [SC96] P. Stocks and D. Carrington. A Framework for Specification-Based Testing. *IEEE Transactions on Software Engineering*, 22(11):777–793, November 1996.
- [Spi92] J. M. Spivey. *The Z notation: a reference manual*. Prentice Hall International (UK) Ltd., Hertfordshire, UK, UK, 1992.