MCC'2017 – The Seventh Model Checking Contest & Marcie

Monika Heiner, Christian Rohr

Brandenburg Technical University, Institute of Computer Science, Cottbus, Germany
{monika.heiner,christian.rohr}@b-tu.de

Abstract. *The following text had been written in September 2017 for an invited section meant to appear as part of an invited paper, later published as [5]* [1]*. The invite came with the titles of the subsections and the constraints: 'TOTAL = 2 pages max + 4 refs max'. However, we hit - somehow - the ceiling of the leading authors, and our text didn't make it into the published paper.*
For those who are interested anyway and want to form their own opinion, we provide here our originally submitted text, extended by three tables, which are also available at https://www-dssz.informatik.tu-cottbus.de/DSSZ/Software/Marcie.

1 Marcie

Marcie [2] offers a variety of qualitative and quantitative analyses including model checking for standard and stochastic Petri nets; the coloured counterparts are supported by unfolding. Particular features are symbolic state space analysis comprising efficient saturation-based state space generation, symbolic evaluation of standard Petri net properties (reversibility, liveness, boundedness), and CTL or CSL model checking.

Most of Marcie's features build on Interval Decision Diagrams (IDDs) to efficiently encode interval logic functions representing marking sets of bounded Petri nets, or to encode the constraints determining the transition instances during unfolding of coloured Petri nets. Additionally, we re-introduced this year Zero-suppressed Binary Decision Diagrams (ZBDDs) as reported in [4] for Marcie's predecessor tool, because ZBDDs occasionally perform better for the special case of 1-bounded Petri nets (such as 'Dining Philosophers'). Marcie's latest public release consists of a sophisticated implementation covering ZBDDs and IDDs that makes heavily use of template meta-programming.

Reported Strengths for 2017. Marcie is among those five tools (out of ten), which hit the upper bound in the estimate of tool confidence (*reliability* for short). It achieved a 100% reliability based on 27,041 test cases, which means a 100% success rate in yielding the correct result when providing a result. The involved seven categories comprise: State Space, Upper Bounds, Reachability Deadlock/Cardinality/Fireability, and CTL Cardinality/Fireability.

[1] updated by adding [5], 04/11/2019.

For comparison, here are Marcie's reliability results in the previous two years (before that, the measure wasn't computed): 2015: 92.52 % based on 19,934 test cases, of which 18,443 were assessed as being correct; 2016: 99,99 % based on 27,364 test cases, of which 27,361 were assessed as being correct. This shows a clear and strictly monotone improvement in Marcie's reliability. Our thanks go to all those tool developers who used Marcie as golden prototype to test their own tools and were kind enough to inform us when they found any discrepancy; a highly appreciated privilege, which we believe to have earned by Marcie's performance in previous MCC rounds.

Handling the 2017 "Surprise" Models. A competition needs a ranking. The MCC ranking could be simply done according to the total number of the delivered correct values of all test cases in a given category, minus a wrong result penalty; let's call it *correct value ranking*. That's easy to understand and rather easy to compute. However, it is somehow considered to be too simplistic for the MCC. So, the correct value points are modulated – firstly by bonus points, which currently make one fifth of the total points a tool can get for a given examination, and a tool *A* could get bonus points even when delivering fewer correct results than tool *B* (as long as there is no wrong result). Secondly, three categories of models are distinguished, and the total points (sum of correct value points and bonus points) are weighted accordingly: *known models* (weight 1), *stripped models* (weight 2), and *surprise models* (weight 6). This weighted sum now permits again a ranking; let's call it *point ranking*. Interestingly, both rankings may differ quite substantially.

In 2017, the results of both rankings coincide for the categories *Reachability Formulas* and *LTL Formulas*, but differ particularly dramatically in the category *CTL Formulas*. With six participating tools, both rankings disagree in five of six positions. Marcie delivers the highest number of correct results (9,115, with 517 more than its successor), but Marcie makes it only on place four in the point ranking (with 585 points behind the tool ranked three).

Category CTL Formulas

correct value ranking	tool	correct values	points	*point ranking*
1	MARCIE	9 115	20 851	4
2	LoLA	8 598	56 429	1
3	ITS-Tools	7 603	21 436	3
4	Tapaal	7 462	42 429	2
5	LTSMin	7 055	16 792	6
6	GreatSPN	6 864	20 091	5

In the *Upper Bound* category, according to the correct value ranking, the first three tools are pretty close – Marcie on the first place has just 81 points more than the tool on the third place, but in the point ranking, there are clear margins between the first three, and Marcie is on place three, with 3,733 points behind the second rank.

Category Upper Bounds

correct value ranking	tool	correct values	points	*point ranking*
1	MARCIE	5726	14 200	3
2	GreatSPN	5709	19 834	1
3	ITS-Tools	5645	17 933	2
4	LoLA	4014	13 894	4
5	LTSMin	3958	9 496	5
6	Tapaal	3088	8 286	6

In the *State Space* category, Marcie is ranked third in the correct value ranking (with 356 points more than the tool ranked fourth), but in the point ranking, Marcie is on place four, with 2,418 points behind the third rank.

Category State Space

correct value ranking	tool	correct values	points	*point ranking*
1	TINA.tedd	1641	17 988	2
2	GreatSPN	1611	20 302	1
3	MARCIE	1499	13 462	4
4	ITS-Tools	1143	15 880	3
5	smart	862	10 688	5
6	LTSMin	792	8 400	6
7	TINA.stif	724	6 402	7
8	Tapaal	477	5 040	8

It doesn't require higher mathematics to see the issue.

Lesson Learned from the Contest. When comparing the performance of those tools drawing their strength from symbolic data structures, our main lesson learnt in this year's round of the MCC is striking: *the specific variant of symbolic data structures apparently doesn't matter so much, what finally decides the order upon crossing the finishing line are the heuristics applied to find the better variable order in reasonable time.*

It is a well known for ages that the efficiency of symbolic data structures heavily depends on a sufficiently good variable ordering. The meaning of 'sufficiently good' can be decided by a tool on its own when trying to crack a test case (meaning to be able to analyse it), but it gets a slightly different meaning when it comes to a competition among tools, where performance is measured under resource constraints.

Amperore and colleagues compare in a very recent paper [1] a couple of sophisticated algorithms (14 to be precise), which follow different heuristics to efficiently determine a good variable order. The study has been performed in the context of the tool GreatSPN. The comparison covers also two algorithms included in Marcie since it made its first public appearance in 2009; all details required for re-implementation are published for quite a while, see [3] for the latest related paper. These two algorithms are among the three which have found to show the best average performance [1]. Good for Marcie, but not good enough for a competition with progressively tight margins between the competitors and overly sophisticated ranking rules.

Among others, Amperore et al. investigate the sensitivity of how well these different heuristic algorithms perform when embedded in one and the same symbolic data structure (here MDD – Multi-way Decision Diagram). Their results reveal a very high sensitivity on the given test cases; with other words, the different heuristics perform particularly well on different subsets of test cases. Consequently, the precise outcome of a MCC also highly depends on the chosen mix of test cases. Which raises the question: How to define a fair, i.e. unbiased and representative mix of test cases?

We asked the authors of [1], whether they have any ideas for which kind of models which kind of variable ordering heuristics works best, and got the answer: we tried our best to find some rules, but in vein. We have no doubts, somebody succeeding in cracking this challenging problem will most certainly be among the winners of any next round of the MCC, of which we do hope to see many.

References

1. Amparore, E., Donatelli, S., Beccuti, M., Garbi, G., Miner, A.: Decision Diagrams for Petri Nets: Which Variable Ordering? In: Proc. of the International Workshop on Petri Nets and Software Engineering (PNSE'17). CEUR Workshop Proceedings, vol. 1846, pp. 31–50. CEUR-WS.org (June 2017), http://ceur-ws.org/Vol-1846/paper3.pdf
2. Heiner, M., Rohr, C., Schwarick, M.: MARCIE - Model checking And Reachability analysis done effiCIEntly. In: Colom, J., Desel, J. (eds.) Proc. PETRI NETS 2013. LNCS, vol. 7927, pp. 389–399. Springer (June 2013), http://link.springer.com/chapter/10.1007/978-3-642-38697-8_21
3. Heiner, M., Rohr, C., Schwarick, M., Tovchigrechko, A.: MARCIE's secrets of efficient model checking. Transactions on Petri Nets and Other Models of Concurrency (ToPNoC) XI, LNCS 9930 pp. 286–296 (2016), http://link.springer.com/chapter/10.1007/978-3-662-53401-4_14
4. Heiner, M., Schwarick, M., Tovchigrechko, A.: DSSZ-MC – A Tool for Symbolic Analysis of Extended Petri Nets. In: Franceschinis, G., Wolf, K. (eds.) Proc. PETRI NETS 2009. LNCS, vol. 5606, pp. 323–332. Springer (June 2009), http://www.springerlink.com/content/g44x170156962645/
5. Kordon, F., Garavel, H., Hillah, L., Paviot-Adet, E., Jezequel, L., Hulin-Hubard, F., Amparore, E., Beccuti, M., Berthomieu, B., Evrard, H., Jensen, P., Botlan, D.L., Liebke, T., Meijer, J., Srba, J., Thierry-Mieg, Y., van de Pol, J., Wolf, K.: MCC'2017 – The Seventh Model Checking Contest. Transactions on Petri Nets and Other Models of Concurrency (ToPNoC) XIII, LNCS 11090 pp. 181–209 (2018)