Feedback for MCS project Part 1 (2016-2017)

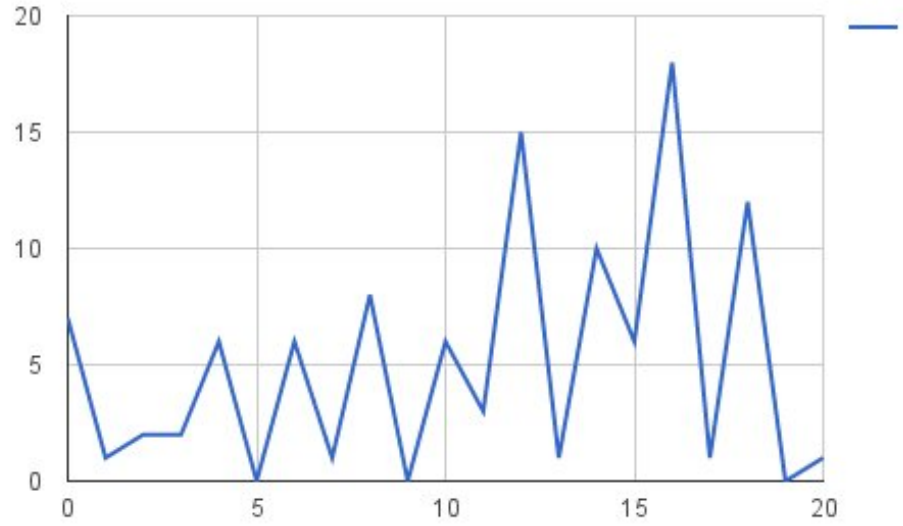
Bart Bogaerts

January 2, 2017

1 Global comments

Happy new year, and best wishes for 2017!

This year, we received 106 submissions for the MCS project. Out of those, 73 passed, 26 failed (with a non-zero score) and 7 received a zero. The average grade on the project was 12/20, with grades ranging from 0 to 20. A distribution of the different grades can be found below.



Grading consisted of three different parts:

- 60% of your score was determined by correctness of the main theory. This part of the project is entirely tested by automated testing procedures.
- 20% of your score was determined by the verification you had to write.
- 20% of your score was determined by quality (readability) of your code.

2 Correctness (60%)

In order to test correctness of the main theory, we created some extra verifications for your theory. These verifications can be classified in two categories: *SAT* and *UNSAT* verifications.

For the *SAT* verifications, we augmented your theory with some extra restrictions to test if specific scenarios are possible by your theory. For instance, theory sat_5 tests if a game can last for at least 5 turns, and theory sat_6 tests if a game can end in a draw. If the corresponding model expansion call returns true, this kind of verifications is successful.

For the *UNSAT* verifications, we tested specific scenarios that should not be possible. Several of them are of the form "the students' definition of predicate_x differs from our definition of predicate_x". For instance the test $unsat\_7$ tests if the predicate "possible" is defined wrongly. Other of these UNSAT verifications test if a given constraint is not respected. For instance $unsat\_1$ tests if a player can pass while there are moves possible.

The final score for this bunch of tests is determined as follows: both for the UNSAT and SAT tests, a score on 6 is calculated: $(6 - \#\{failedtests\})/6$. The lowest of these two scores is taken as the final score. The reason for this type of grading is that the empty theory will always get the maximum score on the SAT tests, but a zero on UNSAT tests, while trivially unsatisfiable theories (for instance, a theory containing a contradiction), always get zero on the UNSAT tests, but the maximum score on the SAT tests. This way of grading ensures that these edge cases get a zero score, while good theories get a high score.

To determine your own score, we attach with this file also the file that runs the automated verifications. You can run it with

idp your_solution.idp checks.idp -e "doourchecks()"

In case your theory fails for a particular UNSAT test, you can print the model that was found to inspect what exactly went wrong. Also, most of these UNSAT tests will still fail if you restrict time to $\{0..1\}$, allowing you to obtain a smaller structure to examine.

2.1 Remarks

- We initially ran this test with a time limit of 3600s and a memory limit of 8gb. For several solutions these limits were not high enough. Since efficiency was not a grading criterion, we then raised the limits. However,

for some solutions this turned out to still not be enough. For these cases, we played around with smaller structures until all tests were covered.

- Some students introduced extra types. This breaks our testing method (and in fact almost any automated testing method). In those rare cases, we individually modified the testing structures to also take an interpretation for the extra type into account and subtracted one point from the verifications.

3 Verifications (20%)

Part of your score was given based on the six verifications you had to write. Model solutions for these verifications can be found in the attached document with our solutions to the project. An important thing to notice is that for some verifications (verifications 1-4), the theory is written in such a way that models of the verification theory are *counterexamples* of the claim. As such, the corresponding procedure takes the form

$if(sat)then$
  $print($"VERIFICATION IS NOT SATISFIED"$)$
$else$
  $print($"VERIFICATION IS SATISFIED"$)$
end

For instance, theory $verification4$ states that *there is a position* for which the claim is false (on which the players play twice). Models of this theory indeed correspond to counterexamples to the claim.

A common mistake is to write the negation of such a theory and check whether there exists a model. However, this way of "verifying" only checks whether it is possible to have a model in which the claim is satisfied: it returns ONE model in which the players do not play twice on the same position, it does not guarantee that this kind of situation can never occur.

The last two verifications were of a different nature. In these two, you had to check that a certain situation is possible. There, the corresponding procedure takes the form:

$if(sat)then$
  $print($"VERIFICATION IS SATISFIED"$)$
$else$
  $print($"VERIFICATION IS NOT SATISFIED"$)$
end

Most students had these last two verifications right.

We subtracted 5% per wrong verification.

4 Readability (20%)

The last bunch of points was given on the basis of quality of your code and documentation. Contrary to the other 80%, this is a subjective criterion. We especially payed attention to documentation of the code. For instance: is it explained in text what certain constraints or definition are intended to mean?

In general we were quite satisfied with this aspect of the project and hence were generous with points here. The average score was 17.5/20; 86 students obtained the maximum score. However, occasionally, we encountered submissions where the comments were restricted to "\\ definition of predicate_x", before each definition or submissions in which auxiliary predicates/types came without intended interpretation. In such cases, it is hard to understand the code and hence, not a lot of points were given to such submissions.

5 Further questions

In principle, we assume that this document, the model solutions, and the automated testing procedures should suffice to make you understand why you obtained a certain score.

In the rare case where this does not suffice, you can send an e-mail to `bart.bogaerts@cs.kuleuven.be`. If you want your e-mail to be taken seriously, please make sure your message clearly shows you have read this document, did an effort to understand where you went wrong (including running the automated tests) and ask specific, directed questions. E-mails simply stating "I believe my solution is good and only got X/20. Why is this?" will live happily in my trash folder for 30 more days.

Good luck with the exams!

Bart Bogaerts
January 2, 2017