CSE 4/586: Project

Name:

Due Date *[2019-12-01 Sun]*

1 Ben Or algorithm

We discussed the Ben Or decentralized consensus algorithm in class. Here is an authoritative resource that describes the algorithm (see Figure 1): `http://disi.unitn.it/~montreso/ds/syllabus/papers/AguileraToeug-CorrecnessBenOr.pdf` You can also refer to a less restrictive version as a secondary supplementary explanation, but not as the authoritative version. `http://www.cs.utexas.edu/users/lorenzo/corsi/cs371d/07F/notes/Week10-Ben-Or.pdf`

You will consider only single instance consensus with Ben Or algorithm.

1.1 Write a PlusCal program to model algorithm in message passing model

Requirements:

- Use TLA+ Paxos style messageboard (`http://muratbuffalo.blogspot.com/2016/11/modeling-paxos-and-flexible-paxos-in.html`), i.e., shared pool messaging:
  - Use p1Msg as the name of the phase1 messageboard
  - Use p2Msg as the name of the phase2 messageboard
  - Messages should be of the form [nodeid, round, value]
- Use the following constants:
  - Use N to denote the number of nodes, and F, such that F<N, to denote the maximum number of processes that may fail.
  - Use MAXROUND to bound the number of rounds a node can try; e.g., use MAXROUND=3 or 4. (Otherwise model checking will not be feasible.)

  - Use INPUT to give initial values to nodes. This is binary consensus, so use 0 or 1 as possible initial values.
  - When you are running your model, in the model overview, you can give values to these constants as follows: `F <- 1, N <- 4, INPUT <- «0,1,1,1», MAXROUND <-1`

- Use the following process local variables (which will be declared/initialized after the line `fair process (p \in Procs)`):
  - Use p1v to denote the value the process will broadcast in phase 1 of a round.
  - Use p2v to denote the value the process will broadcast in phase 2 of a round.
  - Use -1 to denote "$\perp$" or "?" value.
  - Use "decided" variable at each node to store the final/terminal decision a node makes for consensus. Initially *decided=-1*, and only when the process decides, *decided* is set to 0 or 1.
  - So, these variables could be initialized as follows (with r denoting round number of the process): variable r=1, p1v=INPUT[self], p2v=-1, decided=-1;

- You don't need to implement a crash action for any node. We are not in the implementation business, but modeling business. Just having F as the potential failure number is enough, so that nodes don't wait for N replies but only upto N-F replies, and consider any N-F replies as sufficient for going to the next phase.

1.2 Model-check properties with TLA+

Use the toolkit to translate your code to low-level TLA+ code and model-check for correctness.

- Write and test a property to capture the **Agreement** property of the consensus protocol. Agreement should always be satisfied, even when F is more than N/2. Test with N<5 and F $<$ 5, with different values.

- Write and test a property to capture the **Progress** property of the consensus protocol. If you start with all same preference value at all nodes, this should terminate.

- Write and test a **BaitProgress** condition which claims that it is not possible for all processes to decide (*written as a safety property*), and watch the model checker come up with ways progress can happen and show you that consensus can be solved for some runs.

- If N=4 and `INPUT=«0,1,1,1»`, is it possible to have 0 decided for a run? Write a *safety property* called **MinorityReport** which claims that it is not possible for all the nodes to finalize with "0" as the consensus value. The model checker will try to prove you wrong by producing a counterexample when possible. Check this with F=0, F=1, and F=2.

- Write in the comments section, after the `"================"` line, your findings/observations about Agreement, Progress, BaitProgress, and MinorityReport. Support your findings with traces produced by model checker to document runs that violate or satisfy some conditions.

2 Rubric (over 100 points)

- 30% will be awarded for the report you write in the comments, explaining your observations from model checking the above model properties. The grading here will be adjusted based on the correctness of the properties and your model. Having a good explanation of a wrong/incomplete property does not get full mark.

- Agreement property and its correctness checking will be awarded 20%. Agreement should never be violated. Even when F is more than majority of processes, Agreement should still hold. (If you follow the algorithm in Aguilera, this property is satisfied.)

- Progress property and its correctness checking will be awarded 20%. When `INPUT=«1,1,1,1»` or when `INPUT=«0,0,0,0»` Progress should not be violated. That is, every process eventually will have "decided $\neq$ -1".

- BaitProgress property and its correctness checking will be awarded 10%. The model checker should find some computations where all processes will have "decided $\neq$ -1" , even when Progress does not hold (i.e., it is not the case that all computations have "decided $\neq$ -1" for all processes).

- MinorityProgress property and its correctness checking will be awarded 10%.

- Style, i.e., tightness of your model/specifications, will be awarded 10%.

3 Submission

Your TLA+ files should be named *benor.tla*. Your model's name should be the default name $Model\_1$ (do not name your model file differently). Create a zip file from the ".tla" file and the corresponding ".toolbox" directory. **Name the zipfile as: proj.zip**

You will use the submit command ($submit\_cse486$ or $submit\_cse586$ respectively) to submit your work. The submit command instructions are here: `http://wiki.cse.buffalo.edu/services/content/submit-script`

You can have teams upto two people. Include the names and UB-ids of the team members as the first line in the comments section.

The submission deadline is December 1st by the end of the day. **No late submissions will be accepted.** An early submission at 11/20 will receive a 10% bonus points. The bonus points are decreased by one every passing day, and submission at deadline of 12/01 will have zero bonus points.

4 Academic integrity policy

We have zero tolerance on cheating! A copied or plagiarized homework may lead to receiving a failing grade for the course with permanent notation on the transcript that the grade of "F" was assigned for reason of academic dishonesty. Team members are equally responsible. Consult the University Statements on Academic Integrity: `https://engineering.buffalo.edu/computer-science-engineering/information-for-students/policies/academic-integrity.html`

Students who do share their work with others are as responsible for academic dishonesty as the student receiving the material. Students are not to show work to other students, in class or outside the class. Students are responsible for the security of their work and should ensure that printed copies are not left in accessible places, and that file/directory permissions are set to be unreadable to others.

Excuses such as "I was not sure" or "I did not know" will not be accepted. Make sure you go through these questions, and check your answers to these by reading the links on academic integrity.