Proving correctness of MongoRaftReconfig

Ian Dardik

Spring/Summer 2021

1 Intro

This note describes our progress on proving that MongoRaftReconfig satisfies State Machine Safety (SMS) (as well as other properties such as Leader Completeness Safety (LCS) and Log Matching Safety (LMS)). Section 2 talks about our work on proving MongoStaticRaft (the base protocol that resembles Raft) and Section 3 talks about our work on proving MongoRaftReconfig.

2 MongoStaticRaft

2.1 Intro

We have finished proving that MongoStaticRaft satisfies SMS and LCS, and we will talk about the details in this section. I will use the abbreviation MSR to refer to MongoStaticRaft with Log Matching (i.e. MSR = MSRLM); I will explicitly mention "MSR without LMS" if needed.

2.2 MSR Protocol

This is discussed in Will's paper. An additional detail is that MSR is similar to Raft, but not identical. Here are two interesting details of note:

1. MSR can only commit log entries at the end of its log
2. Raft considers any log entries earlier than a commit to be committed. We believe that MSR logically abides by this rule, however the *committed* state variable does not keep track of commits that are prior to a log entry that is explicitly committed by a primary. The *committed* state variable is a ghost variable and hence this does not imply an issue in MSR, however it means that we may be able to improve our safety-property-checking by fixing the "holes" in our ghost variable.

2.3 MSR Safety

The inductive invariant we use is called SMS_LC_II (State Machine Safety, Leader Completeness Inductive Invariant). We have proved that SMS_LC_II 1) implies the initial state of MSR, 2) is inductive invariant on the transition relation for MSR, and 3) implies State Machine Safety. Interestingly, SMS_LC_II does *not* imply Log Matching, and does not require the protocol to satisfy Log Matching to hold. We have proved SMS_LC_II for MSR both with and without Log Matching.

2.4 SMS_LC_II Proof Details

2.4.1 Self Referential Proofs

There are a few places where the proofs rely on an "AndNext" property, i.e. the proof references itself or a previous lemma. I will enumerate the places where I do this and breifly describe why it's safe to do so. While [I believe] I've included the self referential proof steps correctly, the proof would be in better form if I cleaned these up. More coming soon.

2.5 MongoRaftReconfig

2.5.1 Current Status

Proving MongoRaftReconfig (MRR) is a work in progress, the current status is:

1. Will has created an inductive invariant candidate
2. Ian is working on proving that the candidate is correct in TLAPS, 3/23 conjuncts of the inductive invariant are proved so far.

2.5.2 Prep Work

Will devised the inductive invariant candidate for MRR and sent it to Ian and Stavros on 5/13/2021, we will call this inductive invariant candidate MRR_Ind. Ian finished proving SMS_LC_II on 5/31/2021, and started to work on proving MRR_Ind in the next few days. Ian struggled significantly to prove even the first conjunct of MRR_Ind (OnePrimaryPerTerm). The issue is that Ian had not worked with the reconfiguration scheme yet, so Ian built intuition over the next few weeks by trying to find an inductive invariant (using TLC) for MongoLoglessDynamicRaft (MLDR). On 6/24/2021, Ian finally gained enough intuition to prove the first three conjuncts of MRR_Ind. Ian did not find an inductive invariant candidate for MLDR, but this is not important and he will not not continue to look for one.

2.5.3 Max Nat in Finite Subset Issue

TLAPS does not easily prove that there exists a maximum natural in a finite subset of the Nats (which is an obvious result). The issue is that Ian cannot prove that ServersInNewestConfig is nonempty:

```
LEMMA ServersInNewestConfigNotEmpty ==
ASSUME TypeOK, Ind
PROVE ServersInNewestConfig # {}
```

Right now this lemma is left unproven. So far Ian has tried the following two strategies:

1. TLAPS has set up some infrastructure for proving that finite subsets of Nat have a smallest element and the proof is fairly simple. Using a similar technique, I have tried to prove that finite subsets of Nat have a maximum element by:

```
LEMMA ExistsMaxInNatSet ==
ASSUME NEW S \in SUBSET Nat,
       IsFiniteSet(S), S # {}
PROVE \E m \in S : \A x \in S : m >= x
PROOF
    <1>. DEFINE GtR == OpToRel(>,S) \* greater than relation
    <1>1. IsWellFoundedOn(GtR, S) \* unproven
    <1>2. PICK m \in S : \A x \in S : ~ (<<x, m>> \in GtR) BY <1>1, WFMin
    <1>3. \A x \in S : ~(x > m) BY <1>2 DEF OpToRel
    <1>. QED BY <1>3
```

This reduces the proof to step $< 1 > 1$, proving that the "Greater Than" relation is well-founded on the set S. I tried using the same technique as shown in the TLAPM standard library for proving NatLessThanWellFounded (that the "Less Than" relation is well-founded on Nat, the set of naturals). Unfortunately I was not able to make this work; it turns out that copying and pasting the proof itself to prove the original result (NatLessThanWellFounded) doesn't even work. You can find the theorem here: `https://github.com/tlaplus/tlapm/blob/master/library/WellFoundedInduction_proofs.tla`.

2. I looked on the TLA+ Google group and I found a suggestion for the following proof:

```
Max(S) == CHOOSE x \in S : \A y \in S : y <= x

LEMMA MaxIntegers ==
  ASSUME NEW S \in SUBSET Int, S # {}, IsFiniteSet(S)
  PROVE  /\ Max(S) \in S
         /\ \A y \in S : y <= Max(S)
<1>. DEFINE P(T) == T \in SUBSET Int /\ T # {} => \E x \in T : \A y \in T : y <= x
<1>1. P({})
  OBVIOUS
<1>2. ASSUME NEW T, NEW x, P(T), x \notin T
      PROVE  P(T \cup {x})
  <2>. HAVE T \cup {x} \in SUBSET Int
  <2>1. CASE \A y \in T : y <= x
    BY <2>1, Isa
  <2>2. CASE \E y \in T : ~(y <= x)
    <3>. T # {}
      BY <2>2
    <3>1. PICK y \in T : \A z \in T : z <= y
      BY <1>2
    <3>2. x <= y
      BY <2>2, <3>1
    <3>3. QED  BY <3>1, <3>2
  <2>. QED  BY <2>1, <2>2
<1>. HIDE DEF P
<1>3. P(S)  BY <1>1, <1>2, FS_Induction, IsaM("blast")
<1>. QED  BY <1>3, Zenon DEF Max, P
```

Unfortunately I can't get $< 1 > 3$ to work. This code comes from: `https://groups.google.com/g/tlaplus/c/NLLfoXEvxCU/m/K2XgP8S1BAAJ`

Both issues above are likely due to TLAPS version upgrades that broke previous code. I tried looking for updates in the Toolbox but I'm running the latest versions. I'll likely file an issue on the Google group.