Verification of ToyCS Using a Cutoff

Ian Dardik

February 14, 2022

1 Introduction

We introduce the ToyCS protocol and present a key safety property. We then prove protocol correctness using a cutoff proof.

2 ToyCS

ToyCS is encoded in TLA+ as follows:

```
Init == cs = {}

Next ==
    \E p \in ProcSet :
        /\ cs = {}
        /\ cs' = cs \cup {p}

TypeOK == cs \in SUBSET ProcSet

Safety == \A p,q \in cs : p = q
```

The variable *cs* represents the critical section, while *Safety* is effectively mutual exclusion. ToyCS is trivially simple by design. $Safety$ is in fact an inductive invariant itself, and happens to be exactly equal to *Reach*, the set of all reachable states.

3 Verification

3.1 Why Use a Cutoff Proof?

ToyCS and its key safety property are trivial; standard techniques such as model checking and the invariant method can easily be leveraged to verify ToyCS. We will demonstrate correctness using the invariant method–specifically using a cutoff proof to prove consecution–in hopes that eventually we will discover a more general cutoff proof technique that can be automated.

3.2 Cutoff Proofs

There are many different styles of cutoff proofs. In this note we will informally consider a cutoff proof to be an inductive proof on $\mathbb{N}$, where the cutoff is the highest natual that we use in the base case. Thus, the cutoff proof will be a proof for the consecution step in the invariant method; initiation must still be proved in the usual way.

3.3 Initiation

Clearly it is the case that $Init \rightarrow Safety$.

3.4 Consecution

As mentioned in section 3.1, we will use a cutoff proof to establish consecution. We begin by establishing two key lemmas:

Lemma 1. *Let* $S(n) := \{s | s \models Safety(n)\}$. *Then* $\forall n \in \mathbb{N}, S(n+1) = S(n) \cup \{(cs = \{n+1\})\}$.

Proof. Let $n \in \mathbb{N}$ be given. By *TypeOK*, the entire state space is $\{(cs = x) | x \subseteq ProcSet\}$. Now

$$\begin{aligned} S(n) =& \{s | s \models Safety(n)\} \\ =& \{(cs = \emptyset), (cs = \{0\}), ..., (cs = \{n\})\} \end{aligned}$$

Likewise, $S(n+1) = \{(cs = \emptyset), (cs = \{0\}), ..., (cs = \{n+1\})\}$. Hence

$$\begin{aligned} S(n+1) =& \{(cs = \emptyset), (cs = \{0\}), ..., (cs = \{n+1\})\} \\ =& \{(cs = \emptyset), (cs = \{0\}), ..., (cs = \{n\})\} \cup \{(cs = \{n+1\})\} \\ =& S(n) \cup \{(cs = \{n+1\})\} \end{aligned}$$

□

Lemma 2. *Let* $Post_{*n}$ *be short hand for* $Post_{ProcSet=\{n\}}$. *Then* $Post_{*n}(S(n)) = S(n)$.

Proof.

$$\begin{aligned} Post_{*n}(S(n)) =& Post_{*n}(\{(cs = \emptyset), ..., (cs = \{n\})\}) \\ =& \{(cs = \{n\})\} \cup \{(cs = \emptyset), ..., (cs = \{n\})\} \\ =& S(n) \end{aligned}$$

□

Next we present an argument using an inductive cutoff proof to establish consecution.

Lemma 3. *$Safety$ is an inductive invariant for ToyCS, and hence is an inductive invariant for the finite instantiation of each* $n \in \mathbb{N}$. *More precisely,* $\forall n \in \mathbb{N}, Post_n(S(n)) \subseteq S(n)$.

Proof. In the base case, let $n = 0$ and then $Post_n(S(0)) = \{(cs = \emptyset)\} = S(0)$. Now assume that for $n \in \mathbb{N}$, $Post_n(S(n)) \subseteq S(n)$. Then by Lemma 1, Lemma 2, and the inductive hypothesis:

$$\begin{aligned} Post_{n+1}(S(n+1)) =& Post_n(S(n+1)) \cup Post_{*n+1}(S(n+1)) \\ =& Post_n(S(n)) \cup Post_n(\{(cs = \{n+1\})\}) \cup Post_{*n+1}(S(n+1)) \\ =& Post_n(S(n)) \cup Post_n(\{(cs = \{n+1\})\}) \cup S(n+1) \\ \subseteq& S(n+1) \end{aligned}$$

□

Two notes:

1. We only used 0 to establish the base case in Lemma 2, and hence 0 is the cutoff for ToyCS.
2. *Post* is parameterized by *ProcSet*, and hence we write $Post_n$ or $Post_{n+1}$ in the proof.

4 Conclusion

We have verified ToyCS using a cutoff proof during consecusion of the invariant method. Hopefully in the future the proof techniques for a cutoff proof will converge into a more general algorithm or technique to help us verify parametric distributed protocols.