CS 260R, FINAL PROJECT INITIAL WRITEUP

JEFFREY CAI

- **What kind of system are you interested in?** I am looking to verify properties of a randomized hash table.
- **What property are you aiming to verify?** Functional correctness, space bounds, and probabilistic amortized run time bounds.
- **Give a Coq statement of your capstone theorem. This theorem statement need not compile, it may rely on types you haven't defined yet, but it should correspond to the property you want to verify.** This is certainly not correct. The `WithHighProbability` construct is a big hand-wave.

```
insert state k v -> state'
query state k -> state', v
state = (memory, time, potential)

Functional correctness:
k = k' -> query (insert state k v) k' = v
k <> k' -> query (insert state k v) k' = query state k'

Time bounds:
let am_time (memory, time, potential) = time + potential
am_time (fst (query state k)) < am_time state + [a constant]
WithHighProbability, am_time (insert state k v) < am_time state + [a
    constant]

Space bounds:
let mem_used (memory, time, potential) = |memory|
mem_used (fst (query state k)) = mem_used state
mem_used (insert state k v) < mem_used state + [a constant]
```

- **What is your project schedule? What do you aim to have completed each week?**
  - Week 1: Write out algorithm and proofs of the hash table / k-universal hash family by hand. Research any usable prior work on formal verification of randomized algorithms.
  - Week 2: Implement in Coq the word-RAM model in Coq, the hash table algorithm, and the k-universal hash family. Prove the easy (i.e. non-randomized) parts.
  - Week 3: Try to prove the randomized time bound. Extract and run performance tests.
- **What is your division of labor? Whos doing what? How can you work in parallel?** I'm working alone.

Date: April 17, 2017.

- **What are the risks? What are you most worried about in your development? If you cannot prove the capstone theorem, what simpler theorems are you more likely to be able to prove?** The most difficult part by far will be trying to prove the randomized time bound. One could even say that the rest is relatively trivial. In the worst case I would like to prove the other properties and make progress towards proving the randomized time bound.
- **What is your (hypothetical) future work? How could future generations build on your effort?** Many algorithms and systems use hash tables. This work could help to prove efficiency bounds on such algorithms.