State-modeling clipf

Dominic Walden

May 15, 2016

1 License

2 Mission

Using state modeling heuristics to learn about clipf and perhaps find some bugs.

3 Session

While clipf was running I deleted ~/.clipf/. clipf does not recreate file, but nothing bad appears to happen as a result.

What it did reveal, however, is that Operations do not appear to be stored on clipf in the way Products are. After deleting ~/.clipf/ running "op ls" returns nothing, but "prod ls" returns the Products clipf knew about before ~/.clipf/ was deleted.

This suggests that clipf stores Products "internally" and does not re-read the Products database after startup. Operations appear to be stored "externally" and the Operations database is re-read each time an appropriate event is triggered.

If my reasoning above is correct, here are some potential bugs:

1. It means the implementation is inconsistent ("inconsistent with product" in Bolton's terminology). This may or may not concern a user of this program.
2. One of the advantages of using text files might be to allow clipf to run "stateless". This is defeated if Product list is stored internal to clipf.
3. Also, allows the possibility of creating an Operation associated to a non-existent Product, which it normally would not allow.
4. Also, means op and prod commands handle deleting ~/.clipf/ differently.
5. Indeed, clipf thinks there might be products (and be able to add more products) even if product file does not exist at all.
6. Nor does it write to the DB file on shutdown, so consistency is not assured from one session to the next.
7. Doesn't appear to be any way of refreshing the Products clipf knows about (apart from restarting clipf).
8. No recovery from deleting ~/.clipf/.

Attempting to test the "pre-config" state, I changed the permissions on ~/.clipf/ so clipf could not read or write. Started clipf and saw: "Version mismatch db version is 0.3.5 while program require 0.4 version. Run clipf with –migrate option to convert database" Which is hidden functionality which might be worth investigating.

I also discovered that Products are still created (internally to clipf) even if clipf was unable to read to the DB file.

I would like to change the encoding of the DB files to something clipf is not expecting. Implicitly, I am using the "all-the-ways" heuristic as a way of learning about the startup/pre-config state.

There is a way of updating the Products, so there must(?) be a delete event/deleting state.

Potentially interesting states to investigate:

1. Pre-config (Before ~/.clipf/ has been created)
2. Startup (While config. options and databases loaded, ~/.clipf/ created (if applicable), migration to newer DB version)
3. Writing state
4. Reading state
5. Deleting state
6. Shutdown
7. Error handling state