Continuous Integration Testing

The idea behind continuous integration in this context is that everytime a user pushes a commit to github, tests should be ran to verify that nothing has been broken. If anything has been broken by this commit, then there should be a clear indication of a failed build on the relevant commit on github.

1 Compilation

Purpose: Check that compilation of `BLOG` works correctly.

Procedure: I run the following: `sbt/sbt compile`
and use the Unix exit code for determining success.

2 JUnit Tests

Purpose: All JUnit tests located in the `src/test/java` directory pass.

Procedure: Yet to be determined. My goal is to be able to run all the JUnit tests within `src/test/java` on the command line. Options include:

1. Writing a JUnit test suite that lists explicitly all the JUnit files within the directory. This is dispreferred because it would require manual labor and would be difficult to maintain.
2. Dynamically add all the JUnit files to a test suit. Need to explore the Junit add-ons, specifically `DirectorySuiteBuilder`.

3 BLOG Examples

Purpose: All the code examples in `/example` run and return a successful exit code.

Procedure: All the code examples in `/example` are ran using either the shell script `blog` or `dblog`, depending upon the file extension. If any of the examples fail (as indicated by a non-zero exit code), then this section fails. This section only provides a sanity check that all examples run, not that they produce correct output.

4 Incorrect Examples

Purpose: Verify that blog errors are caught and result in non-zero return code.

Procedure: All the files within the directory `tools/error-examples` are BLOG files that should produce a nonzero exit code.

5 Correctness of Examples

Purpose: A single example (e.g. Burglary) with fixed seed produces consistent output that is consistent with the analytic result.

Procedure: Yet to be determined.

6 Integration with Github

Purpose: If any of the tests in sections 1-5 fail, then the current commit should reflect a failed build on github.

Procedure: The `.travis.yml` in the top-level directory of the BLOG project describes the procedure for running the integration code. There is a script, `tools/integration-test.sh` that it calls. If the script returns an exit code of `0`, then the travis build is successful. Otherwise, the travis build fails.