`dilemma` Artifact Evaluation Instructions

Getting Started

Running `dilemma` (without Docker)

The artifact that is run for this evaluation can be found at: https://github.com/ana-brendel/dilemma-artifact. All dependencies and environment to run the tool and evaluation are composed in the docker image (stored a .tar file) contained in this artifact. The artifact and the instructions in this document are for running using Docker (see the next section).

Running Artifact

Download the file `dilemma-docker.tar` that is provided. The following commands will start-up the virtual environment, see the following sections for exact commands to run to evaluate:

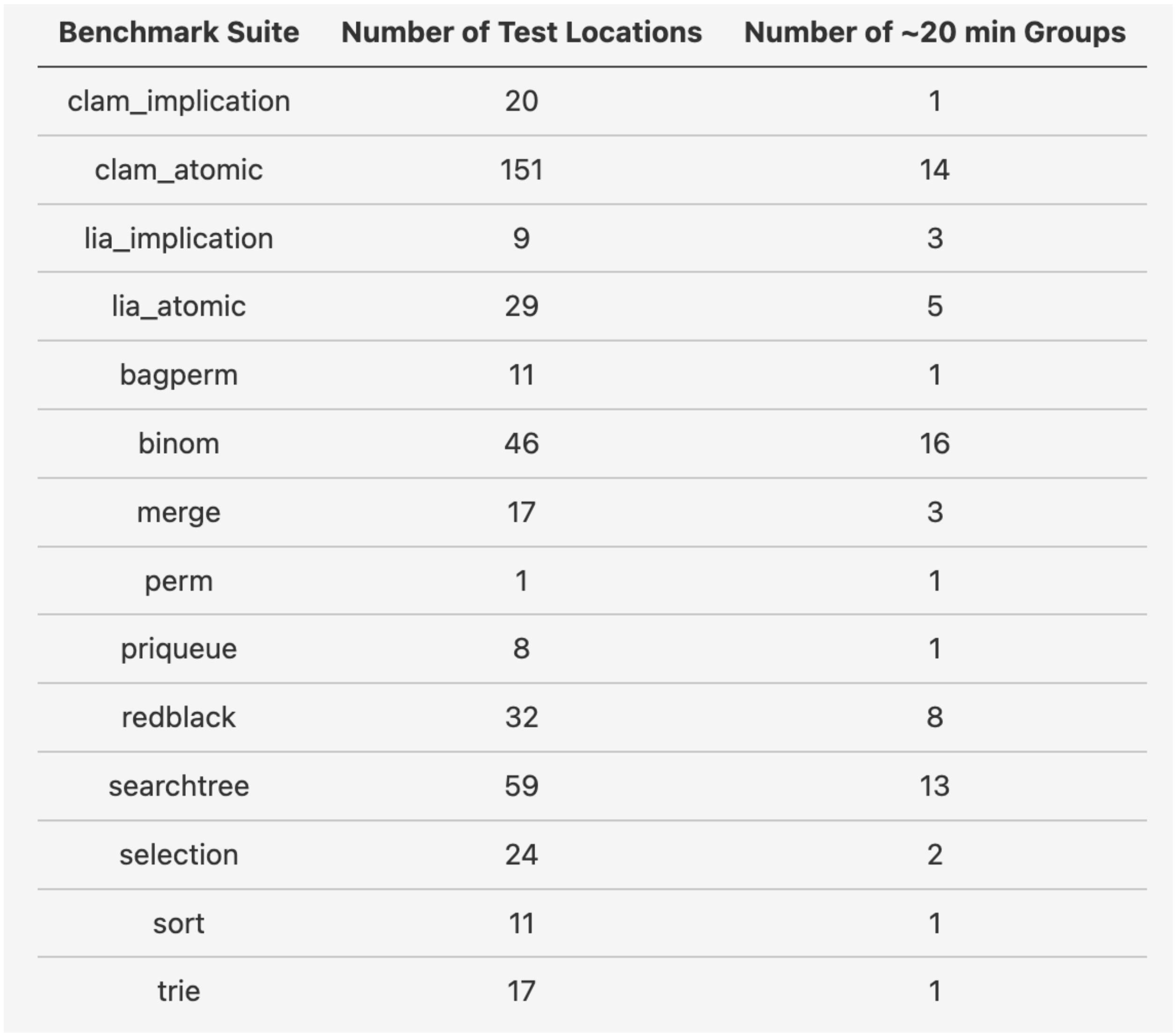
```
$ docker load -i {...}/dilemma-docker.tar
$ docker run -ti -v ${PWD}/results:/root/dilemma-artifact/results
dilemma-artifact
                        <starts environment>
$ root@bf512e7c455b:~/dilemma-artifact# bash make_all.sh
$ root@bf512e7c455b:~/dilemma-artifact# eval $(opam config env)
```

At this point, the environment is set up to run. The "`make_all.sh`" script ensures that all the dependencies for the tests are in place and installed. Finally, "`eval $(opam config env)`" ensures that the command line is able to find all path variables (specifically, for our evaluation that means the command "coqc" is available). The following sections will detail experiment setup and evaluation instructions.

Note, all of the results from the evaluations are expected to be found in the `${PWD}/results` directory as specified in the run command. That is, after running the container there should be a directory created in that same directory titled `'results'`.

Experiment Setup

There are 14 sets of benchmarks that we are currently evaluating on. Based on reviewers feedback and visions, we expect that approximately 2 more sets of benchmarks will be added. The table below details the benchmarks currently accounted for in this artifact:



Benchmark Suite	Number of Test Locations	Number of ~20 min Groups
clam_implication	20	1
clam_atomic	151	14
lia_implication	9	3
lia_atomic	29	5
bagperm	11	1
binom	46	16
merge	17	3
perm	1	1
priqueue	8	1
redblack	32	8
searchtree	59	13
selection	24	2
sort	11	1
trie	17	1

We've provided two means of running the tests. The first option is to run a benchmark suite at a time. In many cases this is probably easiest, however, there are a few larger benchmarks that we have divided into groups to make it easier to run groups at a time and come back later. The rightmost column details how many groups that benchmark was divided into.

Note, that we cut off each group at ~20 minutes, however this was done greedily. So, there are cases where the full 20 minutes is not taken, and in cases where a single test might take longer than 20 minutes that group will just hold that single test. The different groups from each benchmark and their expected runtimes are listed below:

Benchmark Suite	Group Labels	Expected Runtime (minutes)
clam_implication	clam_implication	22
clam_atomic	clam_atomic_1 clam_atomic_2 clam_atomic_3 clam_atomic_4 clam_atomic_5 clam_atomic_6 clam_atomic_7 clam_atomic_8 clam_atomic_9 clam_atomic_10 clam_atomic_11 clam_atomic_12 clam_atomic_13 clam_atomic_14	21 20 15 21 18 21 13 21 16 29 21 21 20 3
lia_implication	lia_implication_1 lia_implication_2 lia_implication_3	16 46 20
lia_atomic	lia_atomic_1 lia_atomic_2 lia_atomic_3 lia_atomic_4 lia_atomic_5	20 20 19 20 7
bagperm	bagperm	12
binom	binom_1 binom_2 binom_3 binom_4 binom_5 binom_6 binom_7 binom_8 binom_9 binom_10 binom_11 binom_12	20 20 20 16 21 19 19 38 18 20 6 17

	binom_13	38
	binom_14	15
	binom_15	29
	binom_16	4
merge	merge_1	16
	merge_2	20
	merge_3	7
perm	perm	1
priqueue	priqueue	18
redblack	redblack_1	11
	redblack_2	22
	redblack_3	16
	redblack_4	20
	redblack_5	21
	redblack_6	15
	redblack_7	18
	redblack_8	19
searchtree	searchtree_1	19
	searchtree_2	19
	searchtree_3	19
	searchtree_4	19
	searchtree_5	6
	searchtree_6	17
	searchtree_7	17
	searchtree_8	13
	searchtree_9	21
	searchtree_10	13
	searchtree_11	19
	searchtree_12	17
	searchtree_13	10
selection	selection_1	20
	selection_2	9
sort	sort	12
trie	trie	18

Each benchmark suite or group contains a set of files, each file is a Coq file which includes a partial proof where the proof is ended with a call to our tactic (`dilemma. Admitted.`). To run each test individually, one can call into the directory and compile the file. For example to run the test select_rest_length_by_select_perm.v from the selection benchmark suite, one would run:

```
$ cd dilemma-artifact/benchmarks/vfa_selection/tests
$ coqc select_rest_length_by_select_perm.v
```

We've set up two means of running sets of benchmarks both using the `run.sh` script: (1) run the whole benchmark suite at a time and (2) run a group at a time (restricted expected runtime). The next section will detail evaluation instructions.

Evaluation Instruction

Generate Results

To run the whole benchmark suite at time run:

```
$ root@bf512e7c455b:~/dilemma-artifact# bash run.sh <benchmark>
```

Where `<benchmark>` is the name of the benchmark suite to run (should correspond to a row in the first table listed in this document). For example, to run the redblack benchmark suite, you would run:

```
$ root@bf512e7c455b:~/dilemma-artifact# bash run.sh redblack
```

The `results` folder will hold the results from this benchmark suite in a file called "`suite_redblack.txt`". The same is expected for running any of the suites (where file is labeled suite_<suite name>.txt).

To run the a single group of benchmarks at time run:

```
$ root@bf512e7c455b:~/dilemma-artifact# bash run.sh group <group>
```

Where `<group>` is the name of the group of benchmarks to run (this should be an element from the middle column of the second table listed above). For example, to run the second group from the `merge` benchmark suite, you would run:

```
$ root@bf512e7c455b:~/dilemma-artifact# bash run.sh group merge_2
```

The `results` folder will hold the results from this benchmark suite in a file called "`group_merge_2.txt`". The same is expected for running any of the groups (where file is labeled group_<group name>.txt).

All results will be generated once all of the benchmark suites have been run. This will either happen by running the benchmark suite as instructed above, or by running all of the groups found within that benchmark suite.

The results generated are the lemmas that we've synthesized – the following section details how these results should be interpreted/evaluated.

Interpret Results

The snip below is from one of the result files:

```
...
--------------------------------------------------------
--------------------------------------------------------
Test: select_rest_length_by_select_perm
Target: select x l = (y, r) -> Permutation (x :: l) (y :: r)

(select x l = (y, r) -> length l = length r)

(select x l = (y, r) -> Permutation (y :: r) (x :: l))
(Permutation (y :: r) (x :: l) -> length l = length r)

(select x l = (y, r) -> Permutation (x :: l) (y :: r))
(Permutation (x :: l) (y :: r) -> length l = length r)

Number of Result Pairs Returned (before QuickChick): 3

Time Elapsed From Start: 58.977 seconds
--------------------------------------------------------
--------------------------------------------------------
...
```

Each result file will have one of these sections for each test that is run. The first line indicates the file that the test is stored in, and the second line represents the helper lemma that was used at the location. The following lines are the lemmas that we've synthesized (this will be restricted to the top 5 for each test). The following two lines are not relevant for evaluation (one notes the amount of results generated and the other is the runtime).

In order to interpret the results, you'll need to manually look at each result. The potential categories a test might fall under are:

- Success – we synthesized the target lemma
    - Look for an exact match to target (variables might differ)
    - Implication has a precondition which is an equality and if that was written in the goal of the implication and that matches the target lemma, then this is also a success. Suppose the target lemma was `Permutation (x :: l) (y ::`

`r)` and we generated `a = x :: l → Permutation a (y :: r)`, this would be counted.

- Useful – we synthesized a useful lemma
    - A lemma found is used later in the same proof, so we found a lemma that was used at some point to complete the proof. This is considered useful. This requires looking at the file to see what lemmas were used later in the proof.
    - Successfully found one precondition, but still need to weaken other assumptions to match exactly. For example, suppose the target lemma is A → B → C and we synthesize A → D → C. This is deemed useful because we've successfully isolated one needed precondition (weakened one assumption).
    - We've found a lemma that is weaker than the target lemma but is able to be used in the same way. For example, if the target lemma is `In x l` ⋁ `In x m → In x (l ++ m)` and we find `In x l → In x (l ++ m)`, where is lemma can be used instead of the stronger target lemma. This is considered useful.
- Fail – test is out of scope, no results generated, or no useful results are generated

We've included the logs from the evaluations that we ran for our paper in the artifact as well. Within the directory `paper_results/processed_results`, there are files for each of the benchmarks containing the categorization. These contain the same information from the results you should have generated (some of the notation is slightly different). These files also include all of the categorization notes that we've included from our analysis.

All of the other folders within `paper_results` hold the raw files that are generated as a byproduct of running our tool. These files should be generated in the docker container as you're running the tests - these will not be moved to the results folder, so if you want to look at them you'll do so through the docker container environment. To complete the evaluation, there shouldn't be a need to look at these files but you can if you are curious.

Expected Results

As cited in our paper, we expect the following breakdown of results…

VFA Benchmarks (226)

- 77 Successes
- 20 Usefuls
- 129 Failures

Lia Benchmarks (38)

- 14 Successes
- 3 Useful
- 21 Failures

Clam Benchmarks (171)

- 60 Successes
- 111 Failures

Note, there is a bit of non-determinism in our procedure for example generation, so it is possible that the results won't match exactly. For example, if insufficient examples were generated, reducing the proof state and/or synthesizing the preconditions might not behave as expected.

Advice for Completing

The total runtime expected to run all of the tests is about 20 hours. We've broken it down so you can do a few chunks at a time and just let it run in the background of whatever you're doing. You can also obviously analyze the results individually before they are all done.

Note, if you run a suite or a group and stop running before it has finished, you won't be able to see the results since they are printed out in batches. This is why we had opted to break the groups down so that you can do it incrementally.

Changes Expected From Revisions

There are two changes that might occur/are expected to occur in the artifact resulting from the revisions suggested by reviewers.

1. More tests will be added. Specifically, we expect that two benchmark suites will be added (each potentially divided into an implication and atomic lemma group).
2. Change to implementation to improve the runtime. Any changes made at this point would be made to improve the runtime and will not be made if any substantial changes to results are caused by the change.